A NEW ADAPTIVE TRILATERAL FILTER FOR IN-LOOP FILTERING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Akitha Kesireddy

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and

Computer Engineering

May 2014

Purdue University

Indianapolis, Indiana

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Kesireddy,Akitha M.S., Purdue University, May 2014. A New Adaptive Trilateral Filter for In-loop Filtering. Major Professor: Mohammed El-Sharkawy.

HEVC has achieved significant coding efficiency improvement beyond existing video coding standard by employing many new coding tools. Deblocking Filter, Sample Adaptivie Offset and Adaptive Loop Filter (ALF) for in-loop filtering are currently introduced for the HEVC standardization. However these filters are implemented in spatial domain despite the fact of temporal correlation within video sequences. To reduce the artifacts and better align object boundaries in video, a new algorithm in in-loop filtering is proposed. The proposed algorithm is implemented in HM-11.0 software. This proposed algorithm allows an average bitrate reduction of about 0.7% and improves the PSNR of the decoded frames by 0.05%, 0.30% and 0.35% in luminance and chroma.

1. INTRODUCTION

High-Efficiency Video Coding (HEVC) is the new video coding standard developed in the joint collaboration of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). The main aim of the project is improving the compression efficiency of the H.264/AVC standard by almost 50% and maintaining the same computational complexity. Many coding tools are included to reduce the distortion between the original frame and decoded frames produced by the lossy coding.

Over the past several years many algorithms have been proposed for reducing the blocking artifacts and the bit rate [2–5].These algorithms can be categorized into several types. One is a post processing algorithm to remove blocking artifacts for highly compressed images in the DCT domain, the other is the reduction of blocking artifacts that is carried out at encoding schemes [2] and another is to reduce the temporal redundancy of ALF parameters by reusing the prior transmitted filter parameters [3]. In [4] the proposed method, a strong filter is selectively applied to blocks having small estimated artifacts that is, to highly reliable blocks to avoid harmful side effects of filtering, and a weak filter is applied to other ones that is, to lowly reliable blocks - in order to slightly correct them. In the other context [5] an adaptive in-loop bilateral filter selecting the optimal filter parameters based on the image characteristics is proposed to minimize the Lagrangian Rate-Distortion. Based on these algorithms we proposed an algorithm which reduces the computational complexity and achieves the main goal to improve the PSNR values and reduce the bit-rate.

1.1 In-loop Filtering

According to the name loop filtering [1] reflects that the filtering is done as a part to remove the blocking artifacts .H.264/AVC includes an in-loop Deblocking Filter. HEVC employs a Deblocking Filter similar to the one used in H.264/AVC but also expands an in-loop processing by introducing two new tools: SAO ad ALF. These techniques are implemented to reduce the distortion introduced in the encoding process (prediction, transform, and quantization). By including filtering techniques, the pictures will serve as better references for motion- compensated prediction since they have less encoding distortion.

1.2 Deblocking Filter

Blocking is known as one of the most visible and objectionable artifacts of block-based compression methods. For this reason, in H.264/AVC, low-pass filters are adaptively applied to block boundaries according to the boundary strength. This improves the subjective and objective quality of the video. HEVC uses an in-loop Deblocking Filter similar to the one used in H.264/AVC. In HEVC, there are several kinds of block boundaries, such as CUs, PUs, and TUs. The set of boundaries that may be filtered in HEVC is the union of all of these boundaries (except for 4 X 4 blocks, which are not filtered to reduce complexity). For each boundary, a decision is made to turn the Deblocking on or off and whether to apply strong or weak filtering. This decision is based on the pixel gradients across the boundary and thresholds derived based on the QP in the blocks.

1.3 Sample Adaptive Offset

SAO [11] is a new coding tool introduced in HEVC, which involves classifying pixels into different categories and adding a simple offset value to each pixel based on its category. SAO classifies reconstructed pixels into different categories based on

either intensity or edge properties. It then adds an offset, either band offset (BO) or edge offset (EO), to the pixels in each category in a region to reduce distortion. BO classifies all pixels of a region into multiple bands, with each band containing pixels in the same intensity interval. The intensity range is divided into 32 equal intervals from zero to the maximum intensity. For example, for 8-b data, the maximum value is 255, so the bands will be $256/32 = 8$ pixels wide.

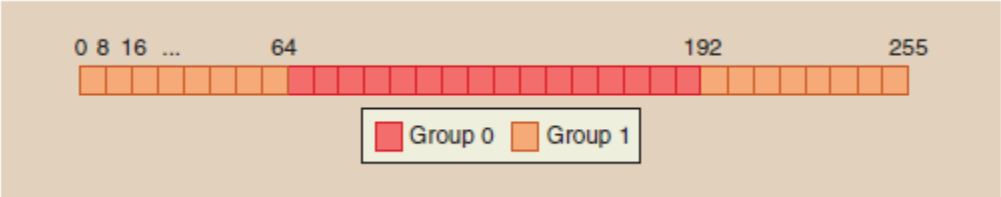

Fig. 1.1: An example of intensity bands and groups of bands in BO mode, for 8-b.

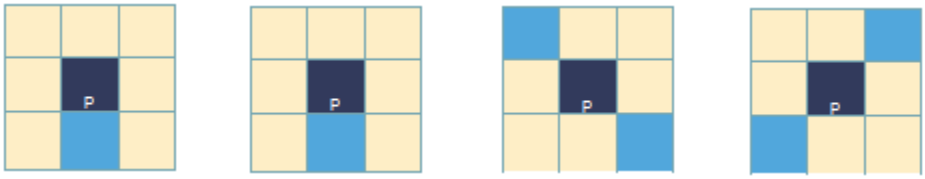

Fig. 1.2: Patterns used in EO mode

The 32 bands are divided into two groups. One group consists of the central 16 bands while the other group consists of the remaining 16 bands (see the example in Figure 1.1). The encoder decides which group of bands to apply SAO, so 16 offsets will be encoded in the bit stream [11]. EO uses one of four one-dimensional three-pixel patterns to classify pixels based on their edge direction, as illustrated in Figure 1.2. Each pixel can be classified as a peak (if it is greater than two neighbors), valley (if it is less than the two neighbors), edge (if it is equal to one neighbor, categories 2 and 3), or none of these. Four offset values will be calculated for these four categories.

The encoder can choose to apply either BO or EO to different regions of a picture. It can also signal that neither BO nor EO is used for a region.

1.4 HEVC Coding Design and Features

The HEVC standard is designed to achieve multiple goals including coding efficiency, ease of transport system integration and data loss resilience, as well as implementability using parallel processing architectures. The following subsections briefly describe the key elements of the design by which these goals are achieved, and the typical encoder operation that would generate a valid bitstream [10].

1.5 Video Coding Layer

The video coding layer of HEVC employs the same hybrid approach (inter/intra picture prediction and 2-D transform coding) used in all video compression standards since H.261. Fig. 1.3 depicts the block diagram of a hybrid video encoder, which could create a bit stream conforming to the HEVC standard. An encoding algorithm producing an HEVC compliant bit stream would typically proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence (and the first picture at each clean random access point into a video sequence) is coded using only intra-picture prediction (that uses some prediction of data spatially from region-to-region within the same picture, but has no dependence on other pictures). For all remaining pictures of a sequence or between random access points, inter picture temporally predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data comprising the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter-picture prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information. The residual signal of the intra-

or inter-picture prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information. The encoder duplicates the decoder processing loop such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization.

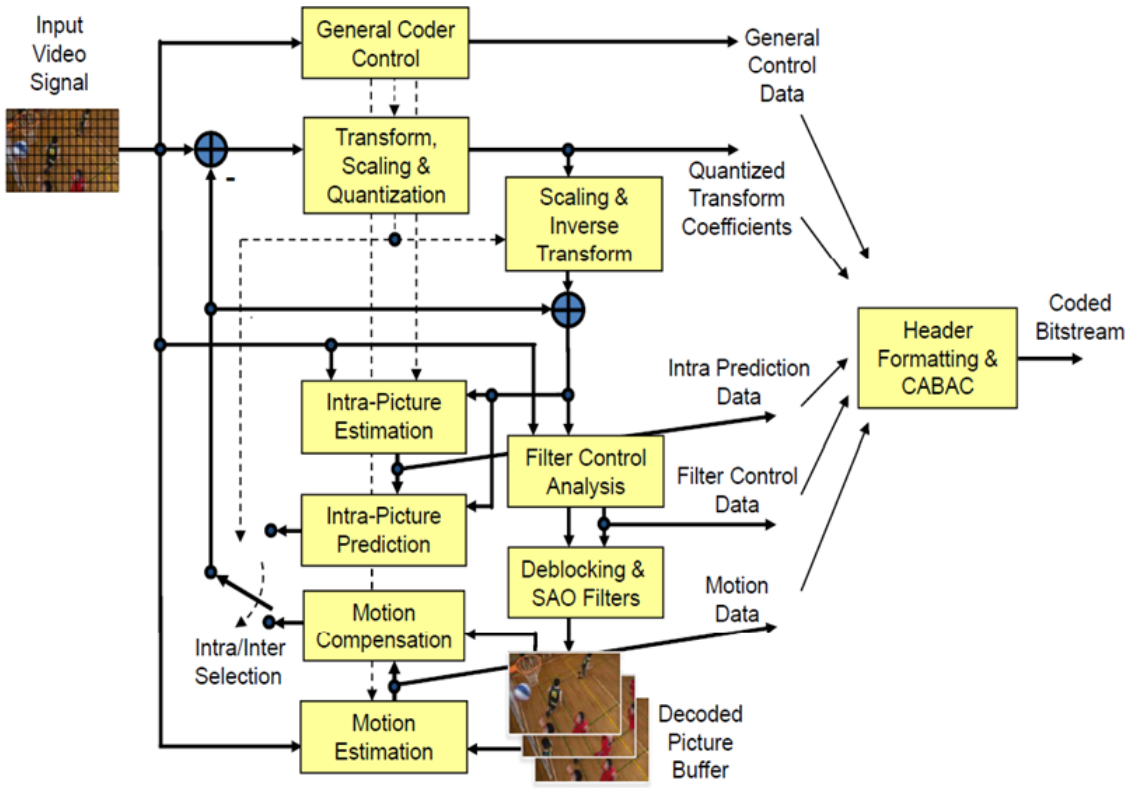

Fig. 1.3: Block diagram of HEVC video encoder

The final picture representation (that is a duplicate of the output of the decoder) is stored in a decoded picture buffer to be used for the prediction of subsequent pictures. In general, the order of encoding or decoding processing of pictures often differs from the order in which they arrive from the source; necessitating a distinction between the decoding order (i.e., bit stream order) and the output order (i.e., display order) for a decoder. Video material to be encoded by HEVC is generally expected to be input as progressive scan imagery (either due to the source video originating in that format or resulting from de-interlacing prior to encoding). No explicit coding features are present in the HEVC design to support the use of interlaced scanning, as interlaced scanning is no longer used for displays and is becoming substantially less common for distribution. However, a meta-data syntax has been provided in HEVC to allow an encoder to indicate that interlace-scanned video has been sent by coding each field (i.e., the even or odd numbered lines of each video frame) of interlaced video as a separate picture or that it has been sent by coding each interlaced frame as an HEVC coded picture. This provides an efficient method of coding interlaced video without burdening decoders with a need to support a special decoding process for it.

Entropy coding: Context adaptive binary arithmetic coding (CABAC) is used for entropy coding. This is similar to the CABAC scheme in H.264/MPEG-4 AVC, but has undergone several improvements to improve its throughput speed (especially for parallel-processing architectures) and its compression performance, and to reduce its context memory requirements.

1.6 Aim of the Thesis

The main aim of this thesis is to reduce the bit rate and improve the video quality by implementing a trilateral filter and adaptive loop filter together, evaluate the effect of proposed algorithm on the quality of the output and compare their results to the evaluated results of the original algorithm of HEVC.

2. DESIGN DESCRIPTION OF FILTERS

2.1 HEVC Adaptive Loop Filtering

This section describes ALF core techniques employed in the final ALF version in HM-5.0. The filter derivation details including Wiener filter derivation, filter shapes, and coefficient coding are introduced here. The concept of Wiener filter is to minimize the mean square error between the desired samples and the filtered samples. In ALF, the desired samples are the original picture, the to-be-filtered samples are the output picture of SAO, and the filtered samples are the output picture of ALF.

In HEVC [1], an ALF is applied to the reconstructed signal after the de-blocking filter and SAO. The filter is adaptive in the sense that the coefficients are signalled in the bit stream and can therefore be designed based on image content and distortion of the reconstructed picture. The filter is used to restore the reconstructed picture such that the mean-squared error between the source picture and the reconstructed picture is minimized. Details of a single filter shape in HM-3.0 which is a cross overlaid on a 3×3 square with nine coefficients to be encoded in the bit stream (Figure 2.1) [13].

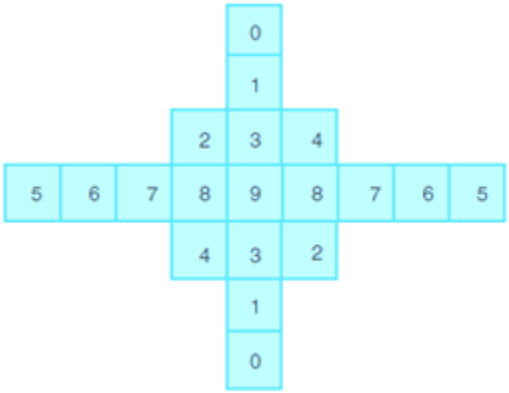

Fig. 2.1: An ALF shape

Note that the number of taps in the filter is greater than nine due to symmetry.

There are two modes that can be used for applying different filters to different pixels within each picture: region-based adaptation (RA) and block-based adaptation (BA). In RA mode, the picture is divided into 16 regions of equal size. These regions can be merged, and each region remaining after merging will have its own filter (with a unique set of coefficients). In BA mode, 4×4 blocks are classified into 1 of 16 categories based on edge activity and direction. These categories can be merged, and in the end, one filter will be designed for each of the categories left after merging. The filter coefficients for each region can be calculated based on the auto correlation and cross-correlation of the original pixels and the reconstruction pixels in the region (using WienerHopf equations) [14].

The ALF can be enabled or disabled for different picture areas based on the partitioning of LCUs into CUs (in a quad-tree segmentation structure).

2.2 ALF Core Techniques

The ALF design in the HM-6.0 and HM-7.0 is detailed in this section which describes ALF core techniques employed in the final ALF version in HM-7.0. The

filter derivation details including Wiener filter derivation, filter shapes, and coefficient coding are introduced from Sections 2.2.1 to 2.2.4. Sections 2.2.5 and 2.2.6 describe related hardware-friendly designs in ALF. Finally ,the flexible syntax design which allows on/off control in different levels is described in Section 2.2.7 [6].

2.2.1 Discrete Wiener Filter

The concept of Wiener filter is to minimize the mean square error between the desired samples and the filtered samples. In ALF, the desired samples are the original picture, the to-be-filtered samples are the output picture of SAO, and the filtered samples are the output picture of ALF.

.

$$Tc = \begin{bmatrix} \sum_{\|R\|} t[r+p_0]t[r+p_0] & \sum_{\|R\|} t[r+p_1]t[r+p_0] & \cdots & \sum_{\|R\|} t[r+p_{N-1}]t[r+p_0] \\ \sum_{\|R\|} t[r+p_0]t[r+p_1] & \sum_{\|R\|} t[r+p_1]t[r+p_1] & \cdots & \sum_{\|R\|} t[r+p_{N-1}]t[r+p_1] \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{\|R\|} t[r+p_0]t[r+p_{N-1}] & \sum_{\|R\|} t[r+p_1]t[r+p_{N-1}]\cdot & & \sum_{\|R\|} t[r+p_{N-1}]t[r+p_{N-1}] \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} \sum_{\|R\|} s[r]t[r+p_0] \\ \sum_{\|R\|} s[r]t[r+p_1] \\ \vdots \\ \sum_{\|R\|} s[r]t[r+p_{N-1}] \end{bmatrix} = v \tag{1}$$

Let us assume the followings for 2-D images.

1. Sample location r = (x,y) belongs to the to-be-filtered region R and $\| R \|$ is the number of samples in R.
2. Desired sample: $s[r]$
3. To-be-filtered sample: $t[r]$
4. FIR filter with N coefficients: $c = [c_0 c_1 ... c_{N-1}]^T$
5. Filter tap position offset:$p_0, p_1, ..., p_{N-1}$, where p_n denotes the sample location offset to r of the nth filter tap.

6. Filtered sample:$f[r]$ and is derived by:

$$f[r] = \sum_{n=0}^{N-1} c_n t[r + p_n] \tag{2}$$

In order to find the minimum sum of squared errors (SSE) between $f[r]$ and $s[r]$,we can calculate the derivatives of SSE with respect to c_n and let the derivatives equal to zero. Then the Wiener-Hopf equations in a matrix form can be derived,as shown in (1). In (1),T, c, and denote the auto-correlation matrix of the to-be-filtered samples, the filter coefficients, and the cross-correlation vector of the to-be-filtered and the original samples, respectively. Gaussian elimination algorithm can be applied to solve (1) for the optimal filter coefficients.

2.2.2 Filter Shape

The filter shapes adopted in the key technical area (KTA)software were 5×5, 7×7 and 9×9 square shapes, where the shape can be selected picture by picture. When ALF was adopted in TMuC, the filter shapes became diamond shapes that reduce the number of coefficients to half. Between TMuC and HM-6.0, filter shapes have been studied by [15–18] to find a good trade-off between coding efficiency and the number of coefficients (i.e., complexity). In HM-7.0, the filter shape of ALF is a combination of 9×7-tap cross shape and 3×3-tap rectangular shape, as illustrated in Fig. 2.2. Each square in Fig. 2.2 corresponds to a sample. Therefore, a total of 19 samples are used to derive a filtered value for the sample of position 9.

2.2.3 Filter Symmetry

Since the Wiener filter is designed according to the coding error statistics of large areas of a picture, the symmetry of coding error is a reasonable assumption. Therefore, a symmetrical filter is used. A point-symmetrical filter can reduce the number

of filter coefficients to half, which implies that the number of multiplications and the overhead sending coefficients can be also reduced to half.Since a multiplier needs much larger chip area than an adder or a subtractor, eliminating half the multipliers significantly reduces the chip area for ALF.

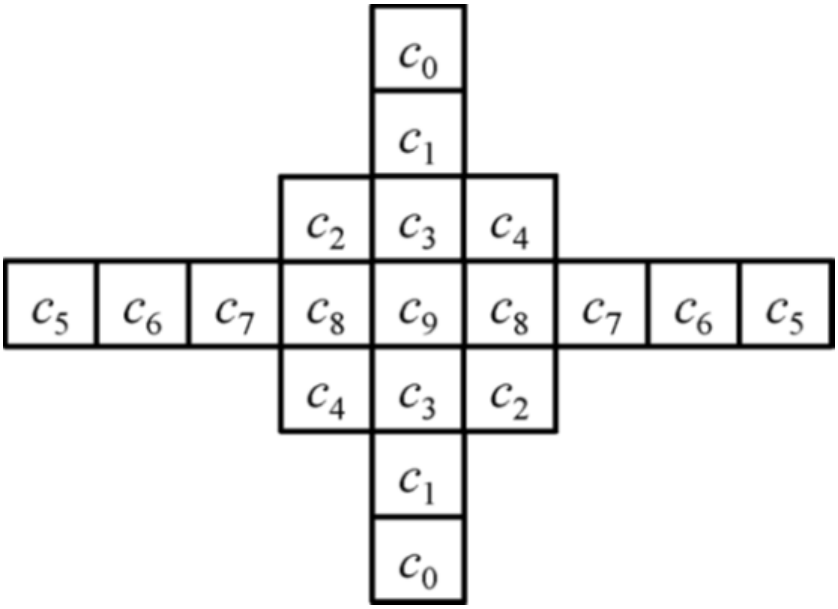

Fig. 2.2: Filter shape of ALF in HM-6.0 and HM-7.0.

In Fig. 2.2, $c_1, c_2, .., c_9$ denote the filter coefficients. Therefore, the number of multiplications is 10 to perform a 19-tap FIR filter.With the symmetric filter design, the Wiener-Hopf equations in (1) can be described in (3) where t' is derived as:

$$t' = [r + p_n] = t[r + p_n] + t[r + p_{18-n}] \tag{4}$$

2.2.4 Filter Coefficient Representation

Filter coefficients are derived by solving Wiener-Hopf equations. The derived filter coefficients can be floating- point, and the filter coefficient ranges are almost unlimited. However, it is hard to represent a variable with a floating-point value and an unlimited range in syntax and implementation, especially for hardware designs [19]. In order to solve this problem, the filter coefficient precision in the fractional part and the filter coefficient range should be properly defined. In HM-6.0, the precision of filter coefficients in the fractional part is 8-bit, so the derived filter coefficients shall be quantized with this precision. The ranges of filter coefficients are defined as follows. For non-center filter coefficients, the coefficient range is limited within $[-1, 1)$. For the center (i.e., position 9 in Fig. 2.2) filter coefficient, the coefficient range is limited in [0.0, 2.0). By these two constraints, filter coefficients can be represented by integers in software implementations, and by 10-bit registers in hardware implementations. The bit widths of multiplier inputs for ALF are 10 bits for filter coefficients and 9 bits for the sum of two samples. The 10 filter coefficients in Fig. 2.2 have different distributions. For example, the center coefficient c_9 usually has the biggest value among all coefficients, and the coefficients near the center position tend to have bigger values than the others.

$$Tc = \begin{bmatrix} \sum_{\|R\|} t'[r+p_0]t'[r+p_0] & \sum_{\|R\|} t'[r+p_1]t'[r+p_0] & \cdots & \sum_{\|R\|} t'[r+p_9]t'[r+p_0] \\ \sum_{\|R\|} t'[r+p_0]t'[r+p_1] & \sum_{\|R\|} t'[r+p_1]t'[r+p_1] & \cdots & \sum_{\|R\|} t'[r+p_9]t'[r+p_1] \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{\|R\|} t'[r+p_0]t'[r+p_9] & \sum_{\|R\|} t'[r+p_1]t'[r+p_{N-1}] \cdot & & \sum_{\|R\|} t'[r+p_9]t'[r+p_9] \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_9 \end{bmatrix} = \begin{bmatrix} \sum_{\|R\|} s[r]t'[r+p_0] \\ \sum_{\|R\|} s[r]t'[r+p_1] \\ \vdots \\ \sum_{\|R\|} s[r]t'[r+p_9] \end{bmatrix} = v \quad (3)$$

Note that the distribution of c_9 can be reduced by a prediction formed by other coefficients using the following equation:

Table 2.1: K Values for Different Filter Coefficients

Filter Coefficient	Coefficient Value Range	K
c_0	-256 to 255	2
c_1	-256 to 255	3
c_2	-256 to 255	3
c_3	-256 to 255	4
c_4	-256 to 255	3
c_5	-256 to 255	1
c_6	-256 to 255	2
c_7	-256 to 255	3
c_8	-256 to 255	4
c_9	0 to 511	1

$$\Delta c_9 = c_9 - (2^8 - \sum_{n=0}^{8} 2c_n) \tag{5}$$

where Δc_9 denotes its prediction error, and 2^8 is the fixed-point representation of 1.0 with 8-bit fractional precision.

To exploit different distributions of coefficients, ALF coefficients are coded using Kth order Exp-Golomb codes. For different coefficients, different K values are selected empirically, as shown in Table 2.1. The coefficient magnitude 0 is signaled by the shortest codeword, the coefficient magnitude 1 by the second shortest codeword, and so on. The sign of each non-zero coefficient is then signaled by one bit. The detailed parsing process of Kth order Exp-Golomb code can be found in [20] for interested readers.

2.2.5 Virtual Boundary Processing

As shown in Fig. 2.2, the filter shape has 9 taps in the horizontal direction, 7 taps in the vertical direction, and 19 taps in total for this filter. This means that the memory bandwidth requirement is 19 times of the number of samples in one picture if there is no local buffer. In hardware implementations, six line buffers can be invested to avoid the memory bandwidth increase. However, the line buffers require large chip areas, especially for high resolution videos. In order to solve this problem, virtual boundary (VB) processing [22] is introduced for ALF. The basic concept of VB processing is that when one sample on one side of a VB is filtered, those samples on the other side of the VB cannot be used in filtering process. That is, if the filter shape centered at a to-be-filtered sample crosses a VB, then the filter process should be changed to avoid using samples on the other side of the VB. The modified filter processes are shown in Fig. 2.3, where c_9 marks the to-be-filtered sample, the bold line in the middle is the VB,$c_0 - c_9$ are original filter coefficients, and $c'_0 - c'_9$ are modified filter coefficients. If the filter shape of the to-be-filtered sample is not across the VB, original filter coefficients are applied in the filtering process. If the filter shape of the to-be-filtered sample is across the VB, the modified filter coefficients are selected based on the position of the to-be-filtered sample and applied in the filtering process as shown in Fig. 2.3.

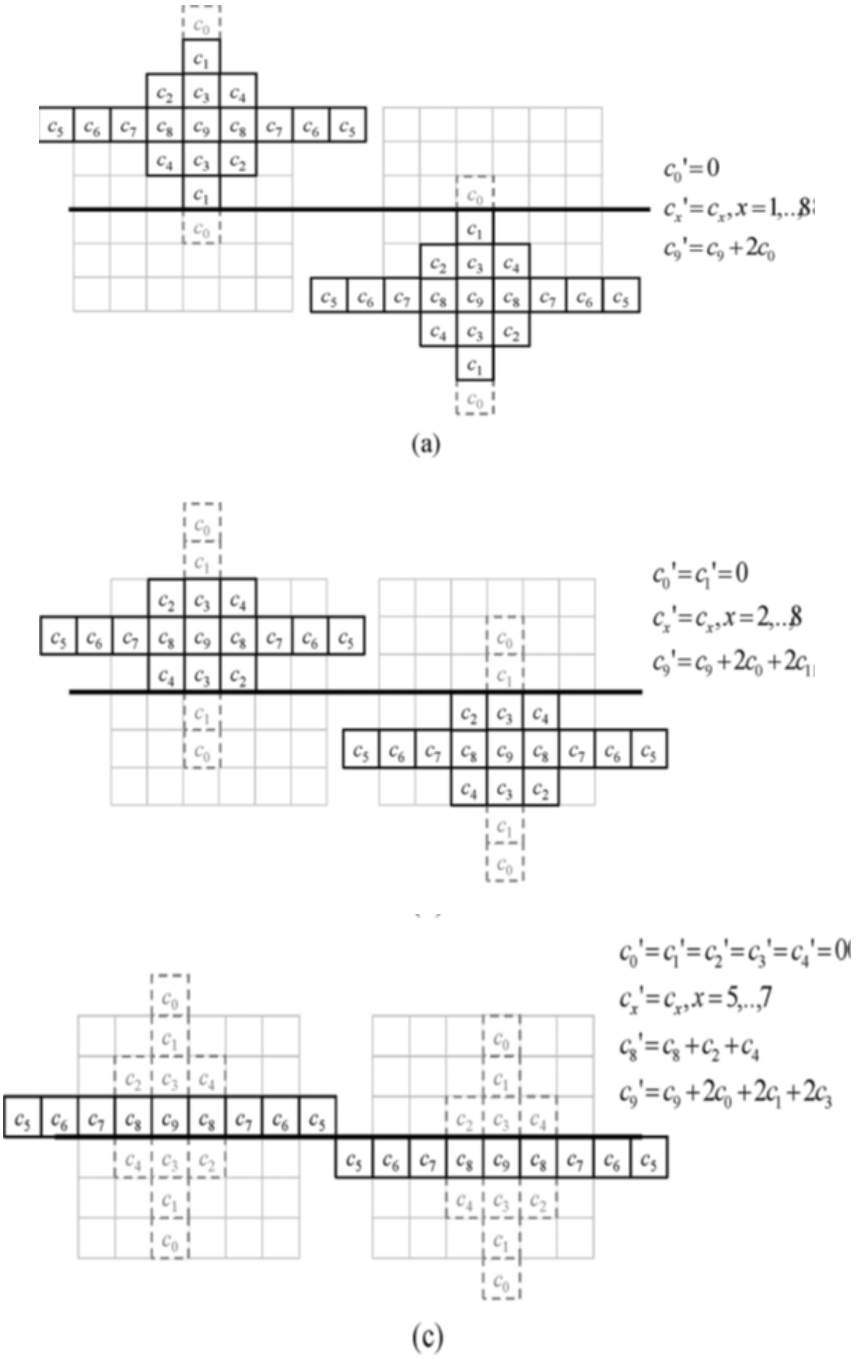

Fig. 2.3: Illustration of virtual boundary processing when filter process across virtual boundary (bold line).

By this method, no sample line buffer is needed in ALF filtering process and the price is only minor coding performance degradation.According to simulation results [22], [23], the BD-rate loss induced by VB process is only 0.1%0.3%, and it is easy to be compensated by increasing the number of vertical filter taps in the filter footprint. Considering the Deblocking Filter in HEVC, the lumaVB is four samples above the horizontal coding tree block (CTB) boundary, and the chroma VB is two samples above the horizontal CTB boundary.

2.2.6 Slice and Tile Boundary Processing

To better support parallel processing, non-cross-slice [24] and non-cross-tile [25] boundary filtering are allowed options. Let us take non-cross-slice boundary filtering as an example. If the filtering process of one sample needs to refer other samples from a different slice, the slice boundary samples of the current slice are repetitively extended to replace those samples outside the slice boundary. The same padding technique is applied for both non-cross-slice and non-cross-tile boundary filtering methods. By using non-cross-slice/tile boundary filtering, the parallel processing can be achieved for different slices/tiles in one picture.

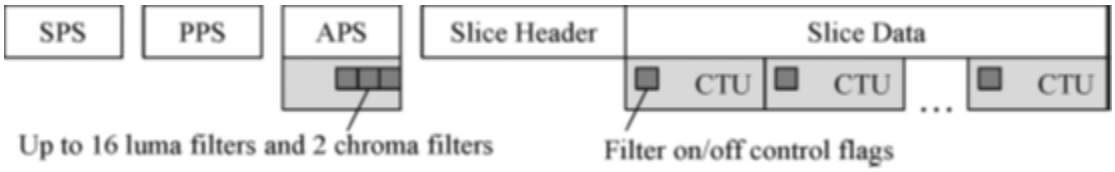

Fig. 2.4: Locations of ALF parameters in the bitstream.

2.2.7 Syntax Design

There are two types of coded information for ALF: filter coefficient parameters and filter on/off control flags. As shown in Fig. 2.4, the filter coefficient parameters

are located in a picture-level header called APS, and the filter on/off control flags are interleaved in slice data with CTUs. The filter coefficient parameters include picture-level on/off control flags for three color components, number of luma filters (i.e., class/region merging syntax elements for BA/RA), and corresponding filter coefficients. Up to 16 luma filters, one Cb filter, and one Cr filter per picture can be signaled. Filter on/off control flags are used to provide better local adaptation. In addition to the picture-level filter on/off control flags in APS, there are also slice-level and CTU-level filter on/off control flags. In slice header, similarly, filter on/off control flags for three color components are coded. The signaling of slice-level filter on/off control flags can solve a slice parsing problem when the referenced APS of the slice is lost [26]. If the slice-level on/off control flag indicates ALF-on, CTU-level filter on/off control flags are interleaved in slice data and coded with CTUs; otherwise, no additional CTU-level filter on/off control flags are coded and all CTUs of the slice are inferred as ALF-off. APS was removed from HEVC standard after HM-8.0. In HM-11.0 to implement ALF without APS, the related syntax elements of filter parameters is coded in slice header instead of APS.

2.3 Block-based and Region-based Filter Adaptation

Block-based adaptation (BA) and region-based adaptation (RA) classify samples according to texture characteristics and sample locations respectively. Details are described in 2.3.1 and 2.3.2. In order to reduce the bit overhead of coding multiple filters, group merge method and syntax support are described in 2.3.3. BA and RA existed simultaneously in HM-5.0, that is, two adaptation methods can be switched to each other in picture level. However, only RA was adopted in the final ALF version in HM-7.0.

2.3.1 Block-Based Filter Adaptation

In order to improve coding efficiency, especially for high resolution videos, a block-based filter adaptation (BA) method is developed to select filters on 4×4 block level, which divides one picture into 4×4 blocks and classifies each 4×4 block based on Laplacian activity and direction. First, vertical and horizontal 1-D Laplacian activities (i.e.,$H_{4\times4}$ and ,$V_{4\times4}$ respectively) for each 4×4 block are computed and only the inner 4 samples are used as shown in Fig.2.5.

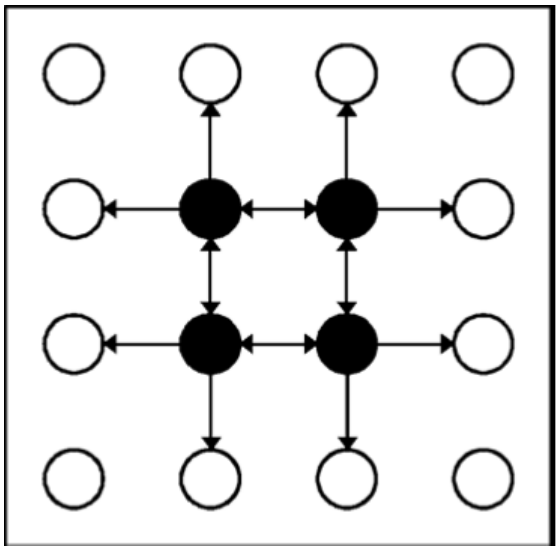

Fig. 2.5: Illustration of samples (solid circles) for calculating a 4×4 block class in block-based filter adaptation (BA) method.

Active Class	0	1	2	3	4	5
Direction=0	0	1	2	3	4	5
Direction=1	0	6	7	8	9	10
Direction=2	0	11	12	13	14	15

Table 2.2: Class Table for Block-based Filter Adaptation

For each 4×4 block with the top-left sample position r,$H_{4\times4}$and $V_{4\times4}$ are given by:

$$H_{4\times4} = \sum_{j=1}^{2}\sum_{i=1}^{2} hor(r,i,j)$$
$$V_{4\times4} = \sum_{j=1}^{2}\sum_{i=1}^{2} ver(r,i,j),$$

where $hor(r,i,j)$ and $ver(r,i,j)$ are derived as:

$$hor(r,i,j) = |2t[r+(i,j)] + t[r+(i-1,j)] - t[r+(i+1,j)]|$$
$$ver(r,i,j) = |2t[r+(i,j)] + t[r+(i,j-1)] - t[r+(i,j+1)]|$$

Then a 2D Laplacian activity is computed by adding $V_{4\times4}$ and $H_4 \times 4$ and quantizing that output into 6 activity classes (i.e., 05). Direction is classified into one of three categories: no direction (0), horizontal direction (1), and vertical direction (2) as follows. If ,$H_{4\times4} \geq 2V_{4\times4}$ direction is 1. Otherwise, if , $V_{4\times4} \geq 2H_{4\times4}$ direction is 2. Otherwise, direction is 0. Based on the 2-D Laplacian activity class and direction, the block based class is derived by using Table 2.2, which results in 16 classes in BA. Note that when there is minor 2D Laplacian activity (i.e., activity class 0), direction information is not used (i.e., block based class 0 is shared for activity class 0 with direction 0/1/2). All 4×4 luma blocks of the same class in a picture share one filter.

2.3.2 Region-Based Filter Adaptation

Besides BA, a region-based filter adaptation (RA) method [21] is also an option to support local adaptation. The concept of RA is to divide one picture into multiple non-overlapping regions, and for each region one local filter can be applied. This adaptation is suitable for one picture with apparent structure and repetitive patterns in one local region.

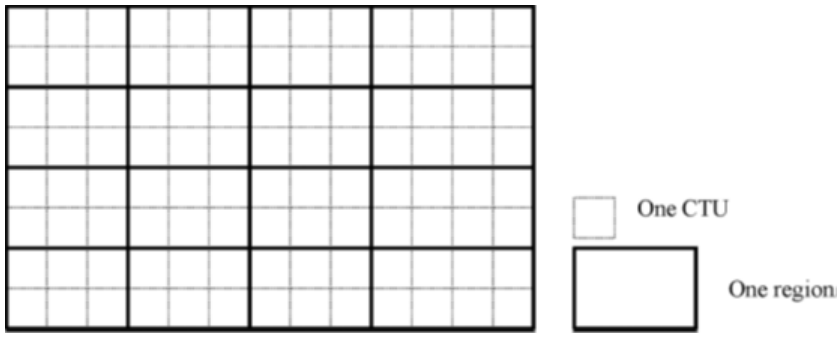

Fig. 2.6: Illustration of dividing one picture into 16 regions in region-based filter adaptation (RA) method.

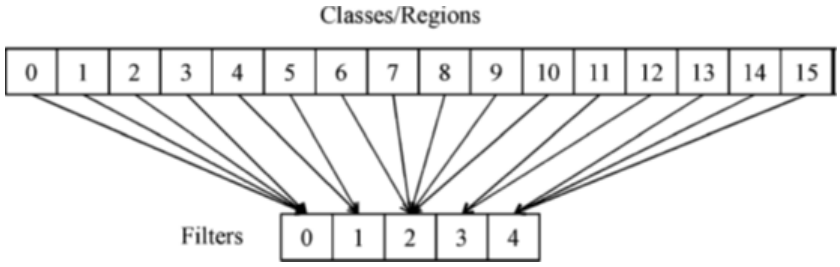

Fig. 2.7: Illustration of samples (Mapping between filters and classes/regions.

For example, one picture is composed of blue sky in the upper part, gray buildings in the middle part, and green grass in the lower part. Compared to BA, RA focuses on the relations of sample position rather than the relations of image texture. It provides another trade-off point between coding efficiency and computation complexity. RA divides one picture into 16 roughly-equal-size regions, as shown in Fig. 2.6. The region boundaries must be CTU boundaries.

2.3.3 Class/Region Merging in BA/RA

In BA/RA, there are 16 classes/regions, and each class/region can have one filter. But depending on bit budget, sometimes neighboring classes/regions need to share

one filter. On the encoder side, a class/region merging algorithm can find the best grouping of classes/regions by trying different versions of merging neighbors based on rate-distortion-optimization (RDO) process. In one extreme, all classes/region share one filter; in the other extreme, each class/region has its own filter. To derive the relation between multiple filters and classes/regions, the mapping information between classes/regions and filters should be signaled to the decoder. A syntax element related to the number of filters is signaled first. This syntax element indicates one of three cases: one filter, two filters, or more than two filters that are used. Fig. 2.7 shows an example when more than two filters are used. In this example, there are 16 classes/regions and five distinct filters the mapping between those can be described as [0, 0, 0, 0, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4], and it can be coded using DPCM coding as [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0]. Note that this mapping information is not needed when one filter or two filters are used for whole classes/regions. When one filter is used, all classes/regions must be merged, so no merging information has to be coded. When two filters are used, the index where the second filter starts to apply is sent.

2.4 Trilateral Filter

An image is defined by f(x) $\in R^n$ (n = dimensionality), where x $\in \Omega$ is the pixel position in image domain Ω.Generally speaking, an n-D (n-dimensional) pixel-discrete image has an image domain defined as,$\phi \subset \Omega \subseteq X_n \subset N^n$(Xn is our maximum discrete index set of the image domain in dimension n). A smoothing operator will reduce an image to a smoothed version of itself, specifically S(f) = s, where s is in the same image domain as f. To introduce the trilateral filter, we must first define the bilateral [8] case; we will then go on to define the traditional trilateral filter using this notation.

The trilateral filter is a " gradient-preserving " filter [8]. It aims at applying a bilateral filter on the current plane of the image signal. The trilateral case only

requires the specification of one parameter σ_1. At first, a bilateral filter is applied on the derivatives of f (i.e., the gradients):

$$g_f(x) = \frac{1}{k_\nabla} \int_\Omega \nabla f(x+a) \cdot w_1(a) \cdot w_2(\parallel \nabla f(x+a) - \nabla f(x) \parallel) da$$
$$k_\nabla(x) = \int_\Omega w_1(a) \cdot w_2(\parallel \nabla f(x+a) - \nabla f(x) \parallel) da.$$

To approximate $\nabla f(x)$,forward differences are used, and more advanced techniques (e.g., Sobel gradients, 5-point stencil) are left for future studies. For the subsequent second bilateral filter, suggested the use of the smoothed gradient $g_f(x)$ instead of $\nabla f(x)$ for estimating an approximating plane:

$$p_f(x,a) = f(x) + g_f(x) \cdot a.$$
$$\text{Let } f_\nabla(x,a) = f(x+a) - p_f(x,a).$$

Furthermore, a neighbour-hood function:

$$N(x+a) = \begin{cases} 1 & \text{if } |g_f(x+a) - g_f(x)| < c \\ 0 & \text{otherwise} \end{cases}$$

is used for the second weighting. Parameter c specifies the adaptive region and is discussed further below. Finally,

$$s(x) = f(x) + \frac{1}{k_\nabla(x)} \int_\Omega \nabla f(x,a) \cdot w_1(a) \cdot w_2(\nabla f(x,a)) \cdot N(x,a) da.$$
$$k_\nabla(x) = \int_\Omega w_1(a) \cdot w_2(\nabla f(x,a)) \cdot N(x,a) da.$$

The smoothed function s equals $S_{TL}(f)$.

Again, w_1 and w_2 are assumed to be Gaussian functions, with standard deviations σ_1 and σ_2, respectively. The method requires specification of parameter σ_1 only, which is at first used to be the diameter of circular neighbour-hoods at x in f; let $\bar{g_f}(x)$ be the mean gradient of f in such a neighbourhood. The parameter for w_2 is defined as follows:

$$\sigma_2 = \beta \cdot |max_{x \in \Omega} \bar{g_f}(x) - min_{x \in \Omega} \bar{g_f}(x)|$$

(β= 0:15 was recommended). Finally, c=σ_2.

3. PROPOSED ALGORITHM

3.1 Proposed Algorithm

There are three in-loop filtering techniques in HEVC; namely, the Deblocking Filtering, the Sample Adaptive Offset (SAO) and the Adaptive Loop Filter (ALF). After the details of these filters in the introduction and the design of the proposed filter the functionality of each filter is noted.

3.2 Boundary Block Detection

In this design trilateral filter can work in the context of block-based processing. The trilateral filter might introduce other blocking artifacts if it is applied to all the blocks in a frame, so it is only applied to blocks in object boundaries. This is called region-based filtering. Here , [9] we use the standard deviation of the block to detect the boundary block since non-boundary blocks usually consist of homogeneous pixel values and have a smaller variance. Only when the standard deviation of a block exceeds a predefined value, we perform the trilateral filtering the standard deviation is for a NXN block is:

$$STD = sqrt\{\tfrac{1}{N\times N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[I(i,j) - Mean_{blk}]^2\}$$

Where N is the block size, I(i,j) is the pixel intensity, and Mean is the mean of the block.

3.3 In-Loop Filtering

After the details of ALF and trilateral filter in chapter 2, it is now essential to define how to combine these filters in the HEVC in-loop filtering process. As described in the introduction, the trilateral filter is a gradient-preserving filter suited to remove the blocking artifacts whereas the Adaptive loop filter is more targeted to reduce the bit-rate. Therefore, it is appropriate to combine these two filters by selecting each image block in the reconstructed frame. This is the main idea behind the proposed algorithm whose processing considered each block along with the Deblocking Filter. The filtering reduces the bit-rate and improves the PSNR values and is not complicated when compared with other algorithms. Now the procedure is detailed in steps by supposing a input frame F into the in-loop filter. The performed steps are:

1. Partition F into block size of B.
2. Using the standard deviation of a block, detect the object boundaries.
3. Over the object boundary perform TLF to obtain F^{TLF}.
4. Perform DBF over the remaining blocks of the frame to obtain F^{DBF}.
5. Finally, the combined frame F^{TDBF} is obtained.
6. Over the whole frame F^{TDBF} perform ALF to obtain F^{TDALF}.

In this algorithm by considering the region characteristics of the block, only the block boundaries are filtered by trilateral filter. Therefore, we adopt the quad-tree structure of LCU in HEVC. For every CUs in LCU, we check whether its standard deviation is above a certain threshold. If the condition is met, we perform the trilateral filtering in this block. Later ALF is performed over the whole frame. The overall flow chart of the proposed in-loop filter for HEVC is shown in Figure 3.1

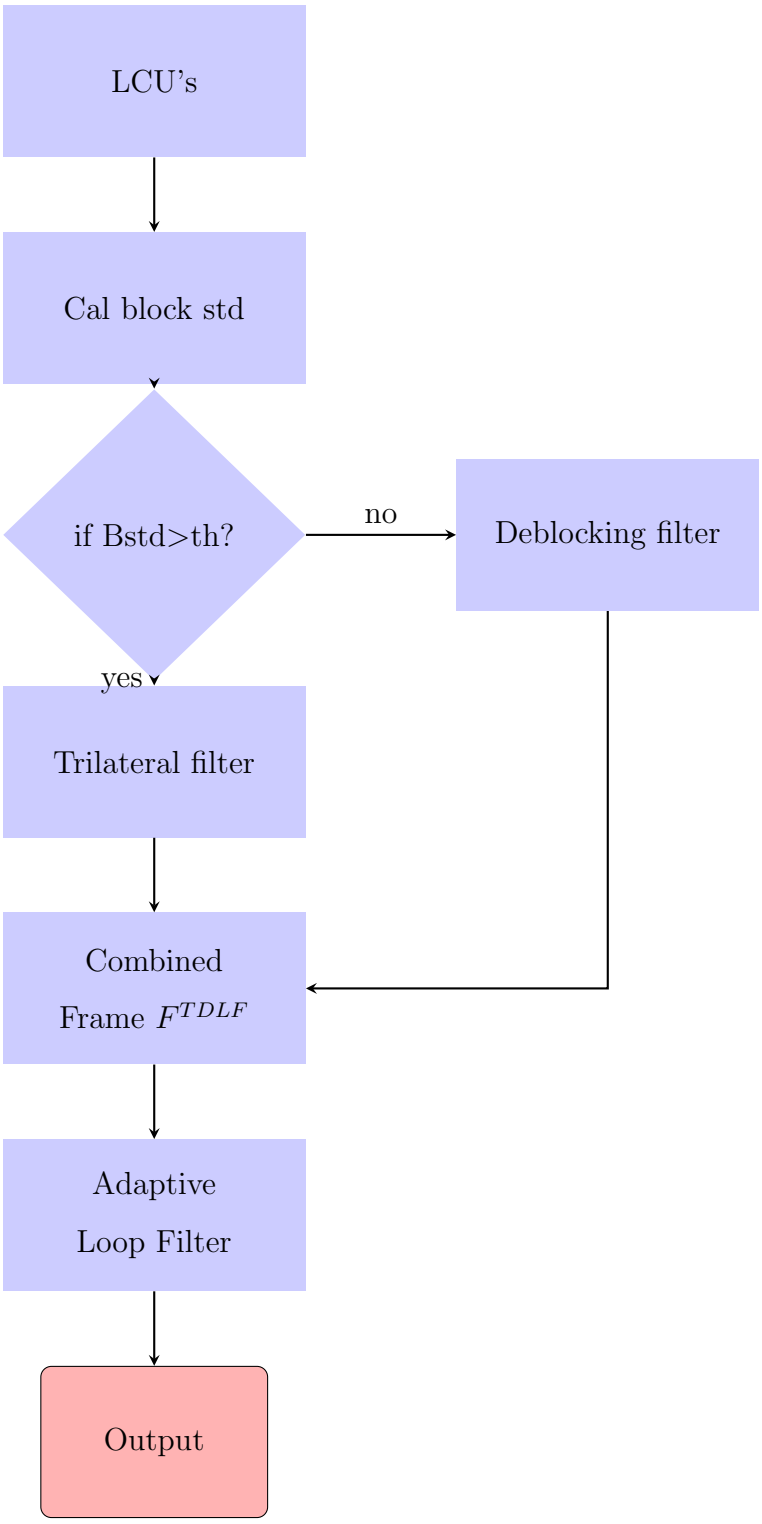

Fig. 3.1: The flow chart of the proposed in-loop filter for HEVC

3.4 Steps to make the HEVC Stream

In this work, the main aim of the thesis is to encode the video with HM-11.0 and produce the HEVC stream. In this process two filters are implemented in the codec and the resulting frames are compared to the frames resulted from the original HEVC Test Model (HM-11.0). HM is provided as a source code so that we can build on various platforms. Windows and Visual Studio Command Prompt (2010) are the tools used. System should have few YUV files as input and YUV Player to view the encoded video. Following the steps below we can encode the video with HM-11.0 [12].

Step 1: Download the source tree

Download the source tree into the latest version of HM download the source tree.

Step 2: Build it!

Start Visual studio command prompt and build HM_vc10 located in trunk/build directory.

```
% msbuild /p:Configuration=Release HM_vc10.sln
```

Step 3:Encode it!

Make a configuration text file and save it as test.config. the encoder tool is called TAppEncoder.exe which is located in Trunk/bin/vc10/win32/release.

```
%TAppEncoder.exe -c test.cfg -i mobile_cif.yuv
```

Step 4:Output in YUV player

Using the above steps we encoded the first 5 frames at 100kbps and it creates mobile.hevc, which is the HEVC stream. The final output is a YUV file which is decoded from the HEVC stream.

Fig. 3.2: Original Image

Fig. 3.3: Encoded Frame

Second figure is the encoded Frame and we can observe some compression artifacts here:

Fig. 3.4: Compression artifacts

4. RESULTS

4.1 Experimental Results

In this paper, we implement the proposed method on HM-11.0 and conduct the same experiments using both the modified HM-11.0 and the original one. For each video sequence varying the quantization parameters say 22, 32, 38 and 42 results are noted in both the cases. 5 frames in the test sequence are encoded. The performance of the proposed method is illustrated under different bases on the resulting subjective image quality and also objective measurement in PSNR. The peak signals to noise ratio show the quality of decoded images, and PSNR is defined as:

$$\text{PSNR} = 10\text{Xlog10 } 1/\text{N X}\sum_{i=1}^{255} \frac{255^2}{(Xi-Yi)}^2 db$$

Where Xi and Yi are the pixel values of the position i of original and reconstructed images respectively. N is the total no. of pixels in the image.

4.2 Proposed Algorithms Quality Assessment Test Conditions

In order to test the efficiency of the proposed algorithm, evaluation of the quality of the reconstructed video compared to the original video is done using the quality assessment mentioned in section 4.1. The tests done in this chapter are done using standard high definition video quality assessment sequences available on [12]. The full description of the test sequences used is shown in table 4.1.

Table 4.1: Test Sequences Information

Sequence	#Frames	Short Description
Mobile_cif	300	A toy train moving on the track pushing the ball forward with the fixed background of a calendar and a painting of scenery with many details.
Akiyo_cif	300	News reporter reading the news with the background of screens telecasting the news. Static camera.
Flower_cif	300	View of a windmill with a good scenic view, high contrast, small color differences in the sky, many details.
Foreman_cif	300	A man explaining the construction view. Camera zoomed in, high depth of the field.
Bus_cif	300	Video capture of a fast moving bus , low camera position, bus pass by very close to the camera.

The aim is to evaluate the complexity of the algorithms compared to their quality and reliability, and determine the efficiency of each of the proposed algorithms compared to the original. The sequences used in the testing are taken as input in the YUV format, which means that each frame is represented in terms of three components; Y which is the Luminance; U which is the Contrast; and V which is the Structure. The sequence is encoded as a set of frames, each frame is treated as an independent image, and each component within each frame is also encoded independently. The same is done for the decoding process after which the components are combined to reform the frames which are combined together to reform the sequence.

4.3 Proposed algorithm experimental results

The test results for each sequence is shown in the rate-distortion graph. For each sequence we have 4 graphs for the PSNR, each figure with a separate graph for each component and each figure with a graph for average value for the proposed algorithm and the original algorithm. In these graphs x-axis is the bit-rate and the y-axis varies as Y,U,V and average PSNR values.

4.3.1 Test Results for the Mobile_Cif Sequence

The results for the tests performed on the sequence [Mobile_cif] are shown from Figure 4.1 to Figure 4.4.

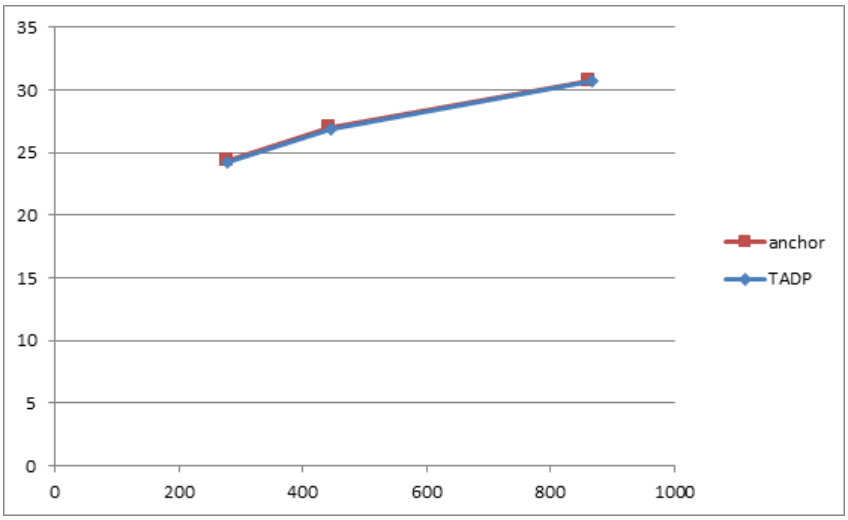

Fig. 4.1: Bit-rate vs Y-PSNR (Mobile_cif)

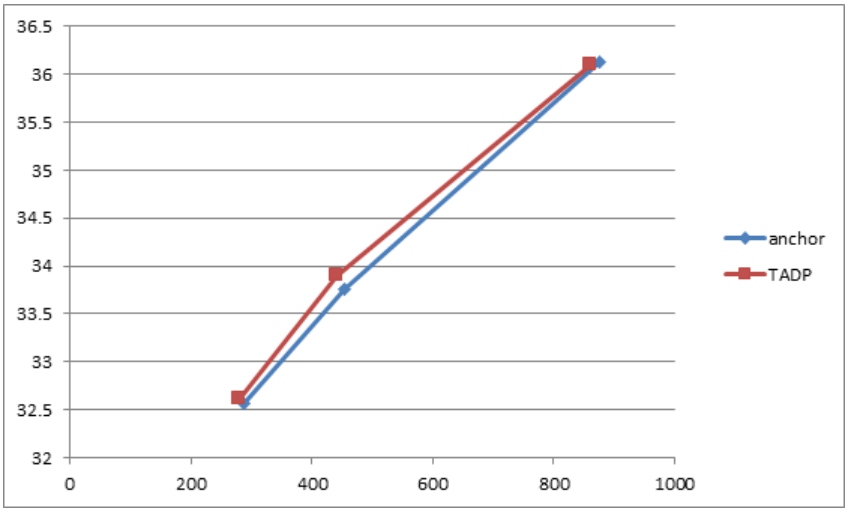

Fig. 4.2: Bit-rate vs U-PSNR (Mobile_cif)

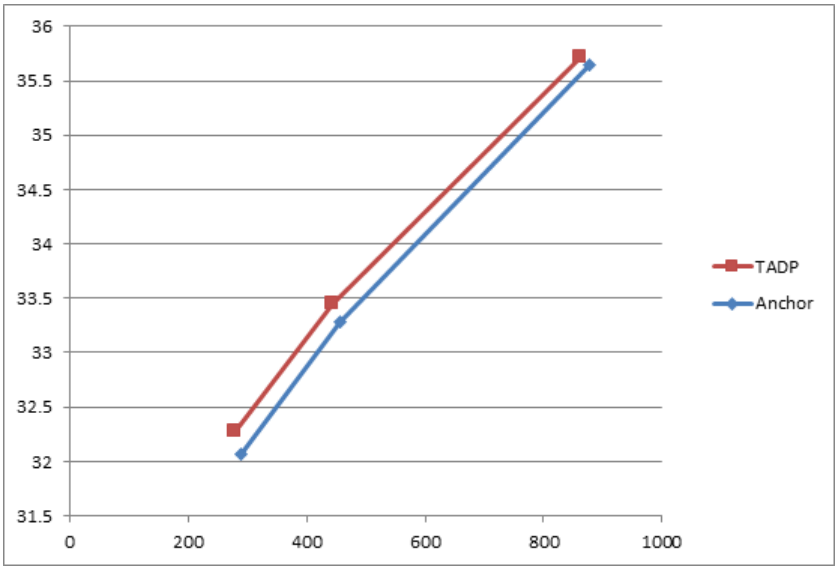

Fig. 4.3: Bit-rate Vs V-PSNR (Mobile_cif)

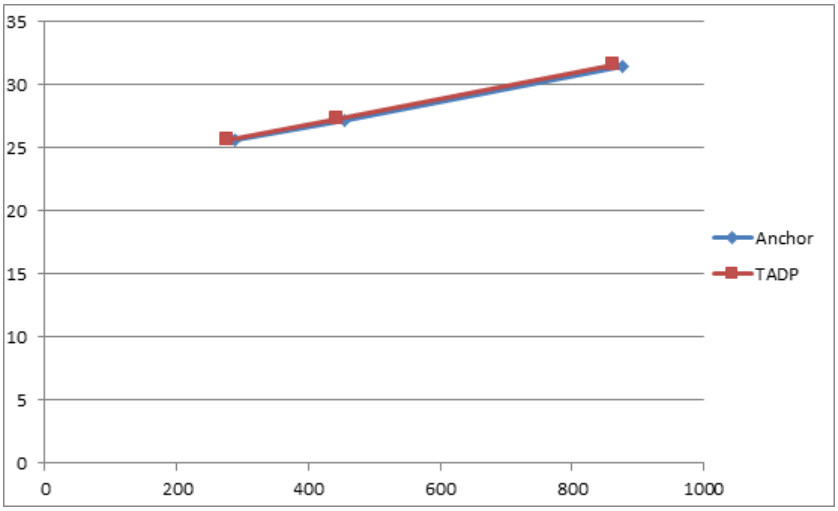

Fig. 4.4: Bit-rate Vs Avg-PSNR (Mobile_cif)

4.3.2 Test Results for the Akiyo_Cif Sequence

The results for the tests performed on the sequence [Akiyo_cif] are shown from Figure 4.5 to Figure 4.8.

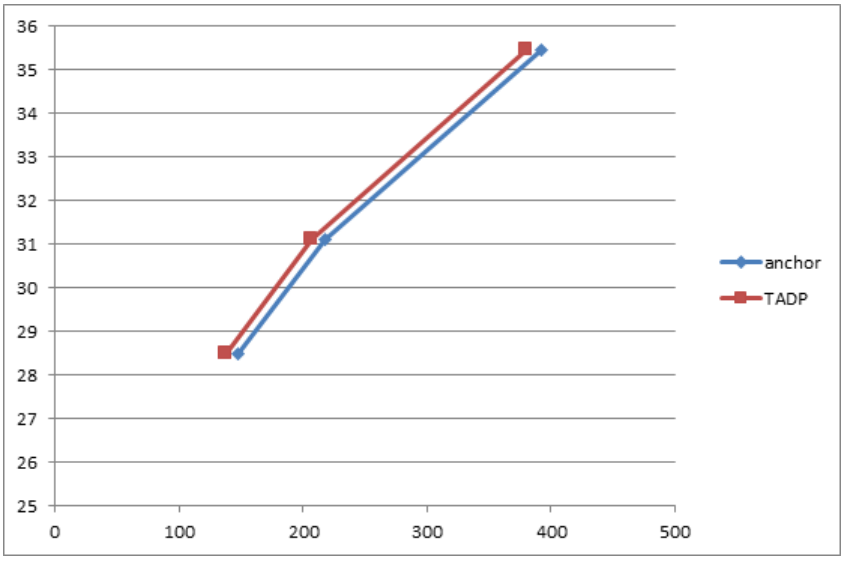

Fig. 4.5: Bit-rate Vs Y-PSNR (Akiyo_cif)

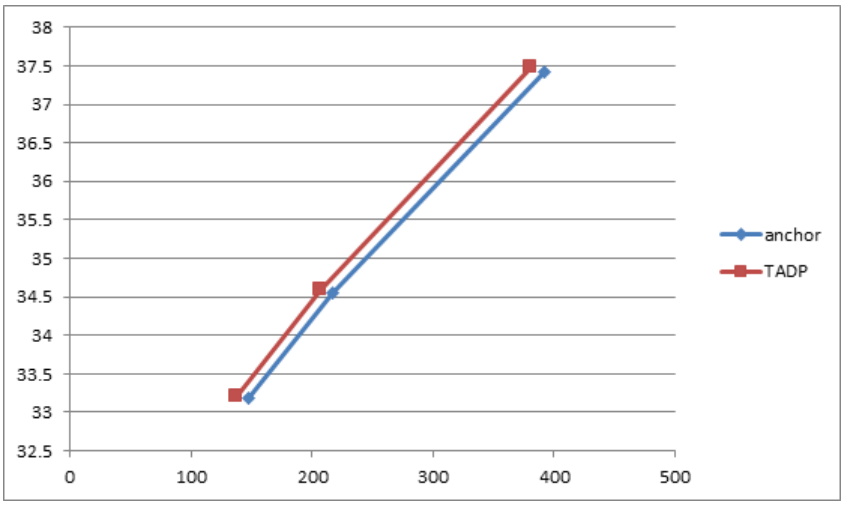

Fig. 4.6: Bit-rate Vs U-PSNR (Akiyo_cif)

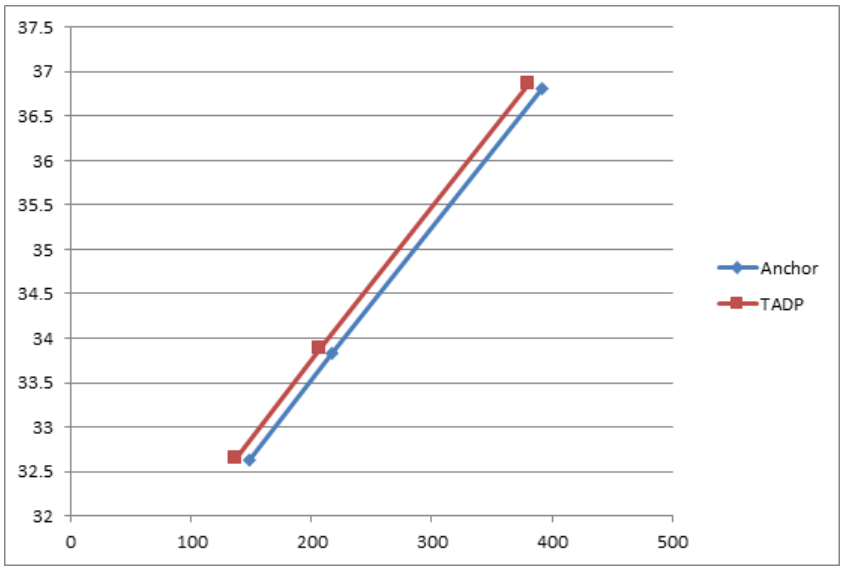

Fig. 4.7: Bit-rate Vs V-PSNR (Akiyo_cif)

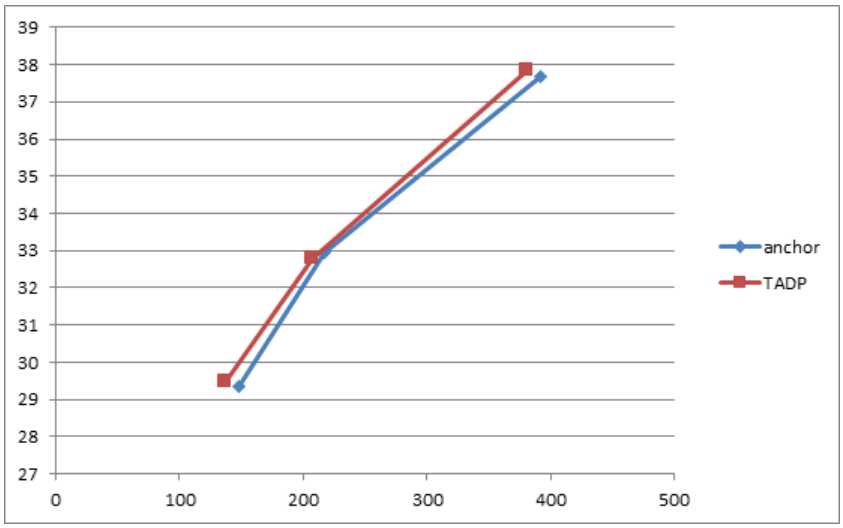

Fig. 4.8: Bit-rate Vs Avg-PSNR (Akiyo_cif)

4.3.3 Test Results for the Flower_Cif Sequence

The results for the tests performed on the sequence [Flower_cif] are shown from Figure 4.9 to Figure 4.12.

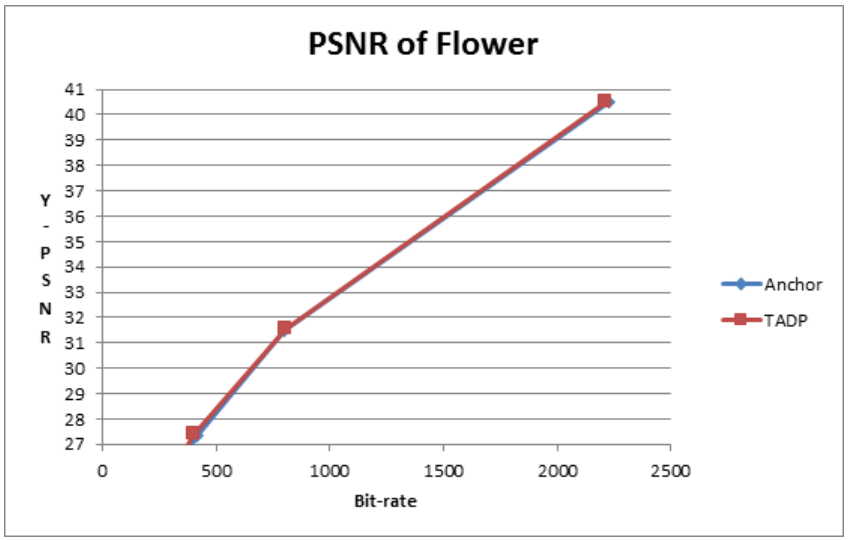

Fig. 4.9: Bit-rate Vs Y-PSNR (Flower)

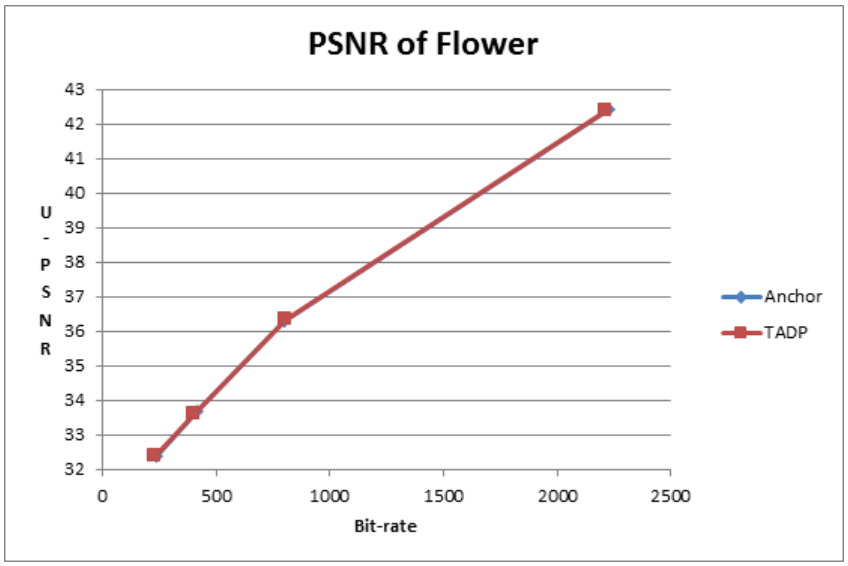

Fig. 4.10: Bit-rate Vs U-PSNR (Flower_cif)

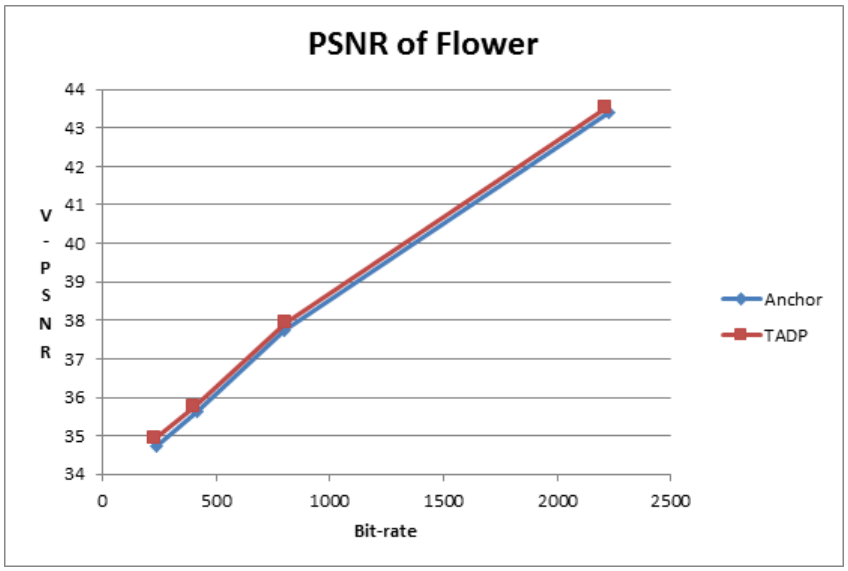

Fig. 4.11: Bit-rate Vs V-PSNR (Flower_cif)

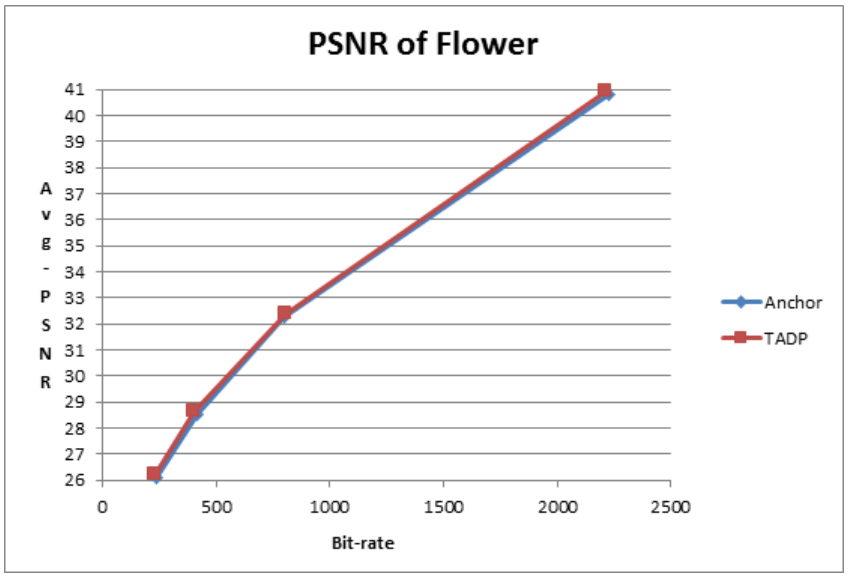

Fig. 4.12: Bit-rate Vs Avg-PSNR (Flower_cif)

4.3.4 Test Results for the Foreman_Cif Sequence

The results for the tests performed on the sequence [Foreman_cif] are shown from Figure 4.13 to Figure 4.16.

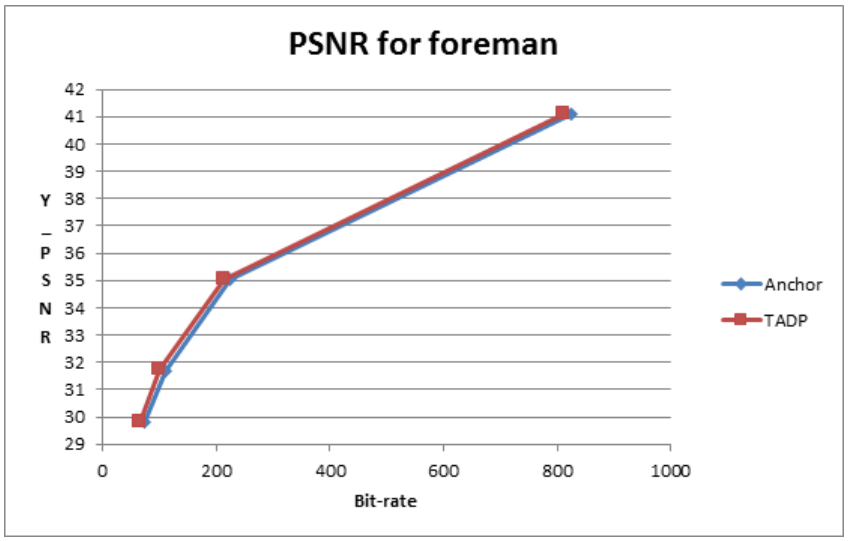

Fig. 4.13: Bit-rate Vs Y-PSNR (Foreman_cif)

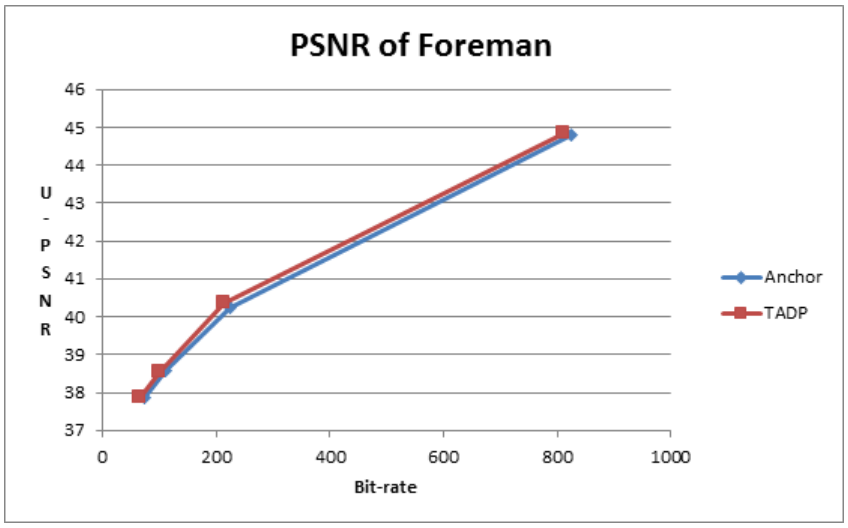

Fig. 4.14: Bit-rate Vs U-PSNR (Foreman_cif)

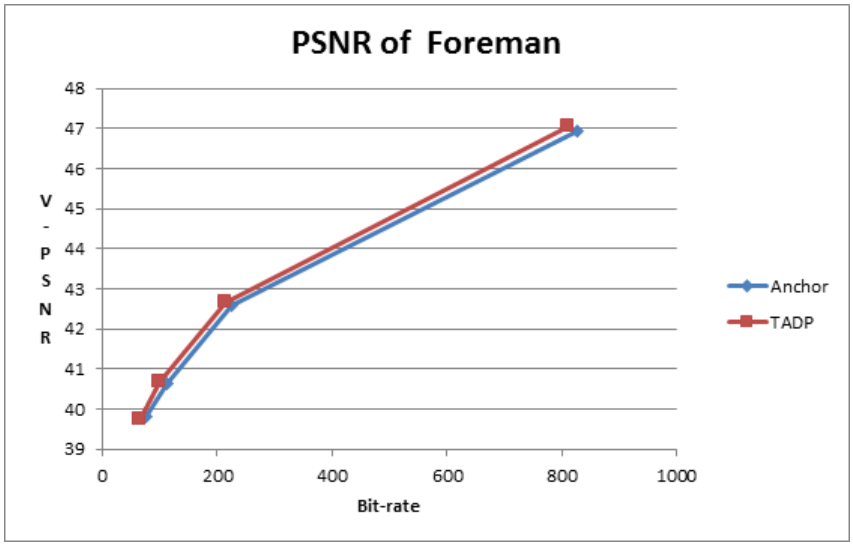

Fig. 4.15: Bit-rate Vs V-PSNR (Foreman_cif)

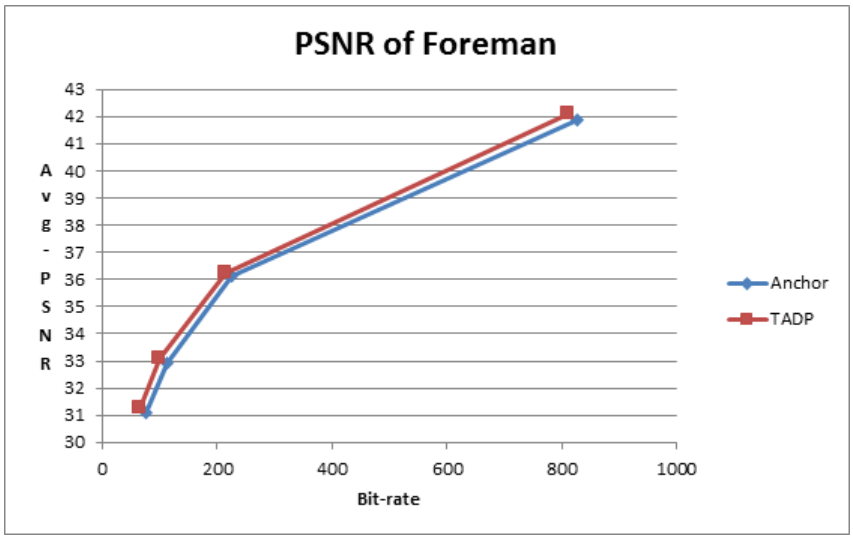

Fig. 4.16: Bit-rate Vs Avg-PSNR Foreman_cif)

4.3.5 Test Results for the bus_Cif Sequence

The results for the tests performed on the sequence [bus_cif] are shown from Figure 4.17 to Figure 4.20.

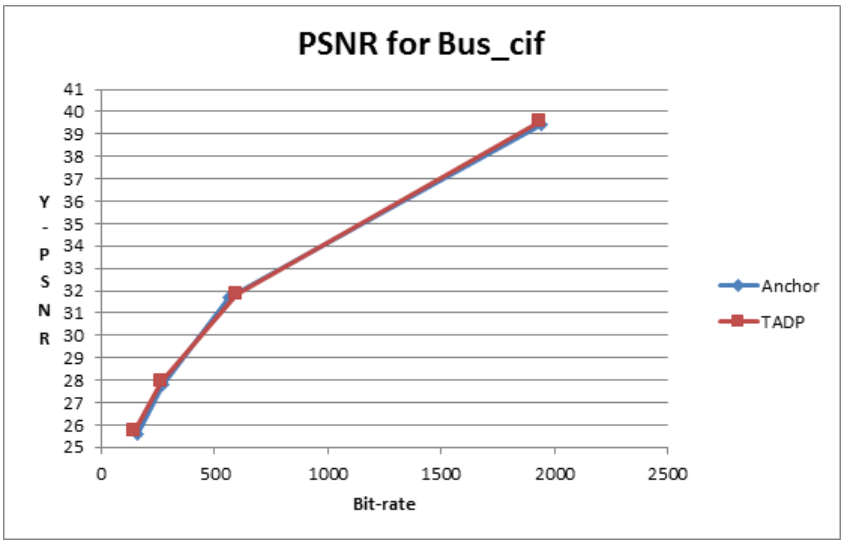

Fig. 4.17: Bit-rate Vs Y-PSNR (Bus_cif)

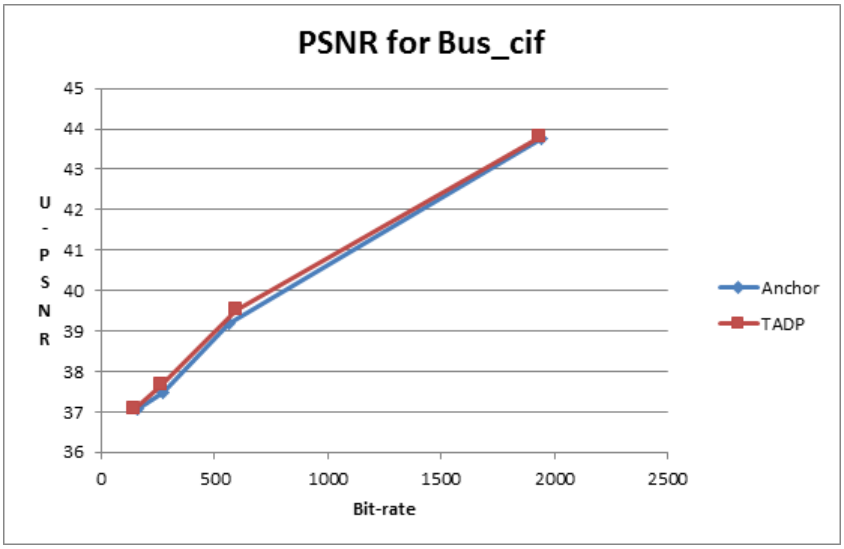

Fig. 4.18: Bit-rate Vs U-PSNR (Bus_cif)

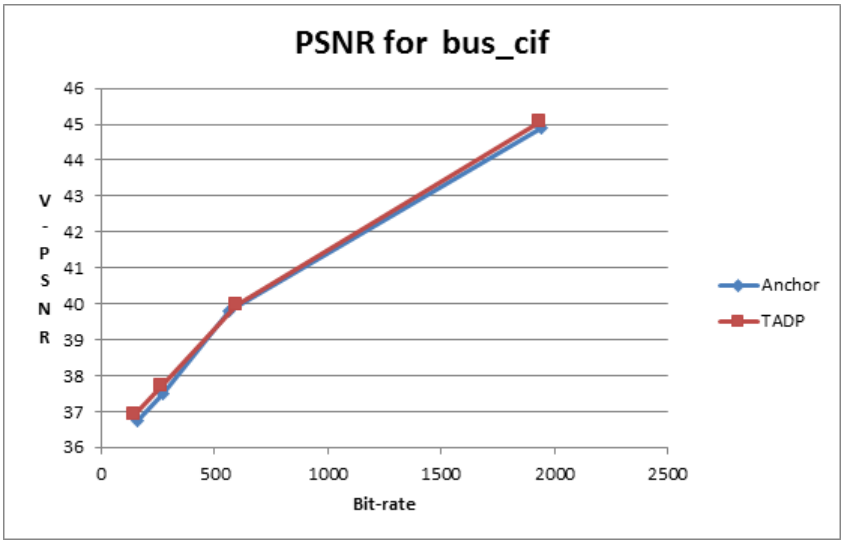

Fig. 4.19: Bit-rate Vs V-PSNR (Bus_cif)

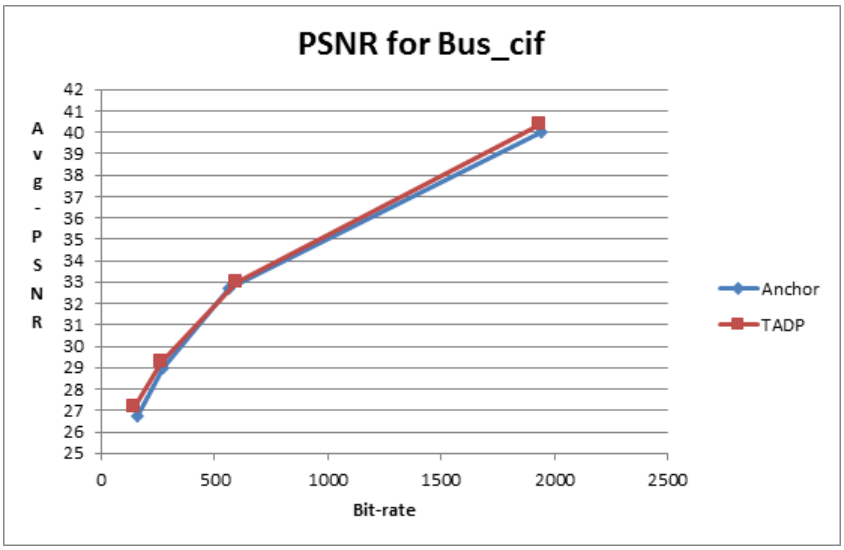

Fig. 4.20: Bit-rate Vs Avg-PSNR (Bus_cif)

4.3.6 Bit-rate changes for the test sequences

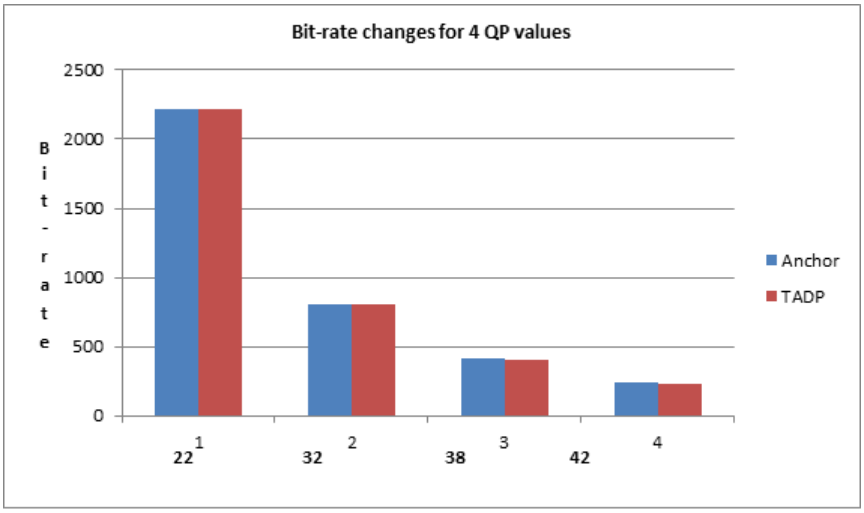

Fig. 4.21: Comparison of the Bit-rate of Flower_cif for different QP values

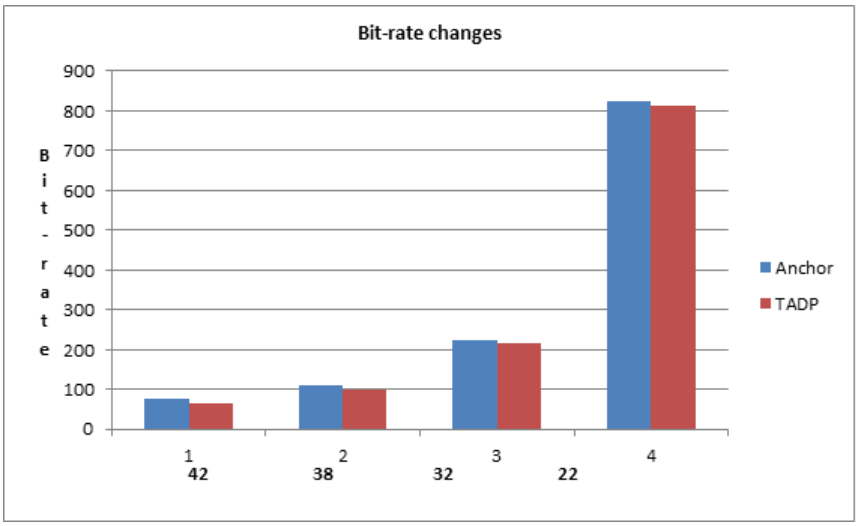

Fig. 4.22: Comparison of the Bit-rate of Foreman_cif for different QP values

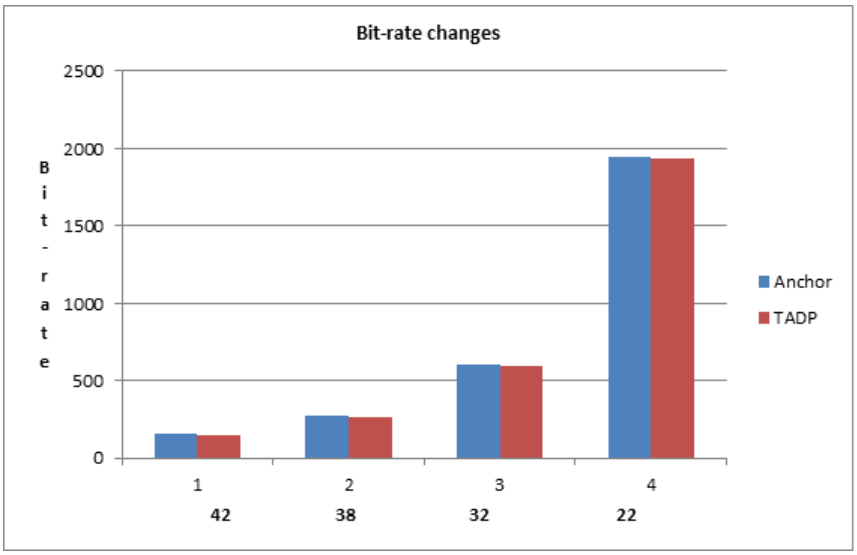

Fig. 4.23: Comparison of the Bit-rate of Bus_cif for different QP values

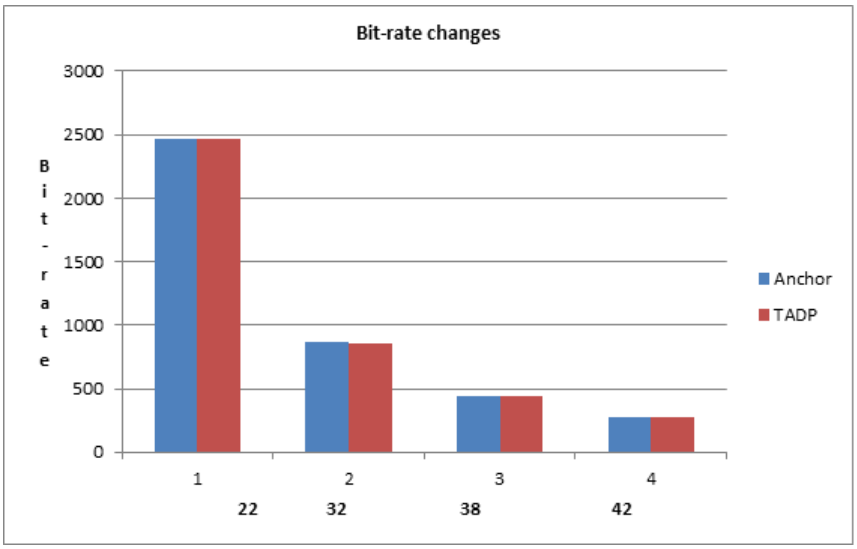

Fig. 4.24: Comparison of the Bit-rate of Mobile_cif for different QP values

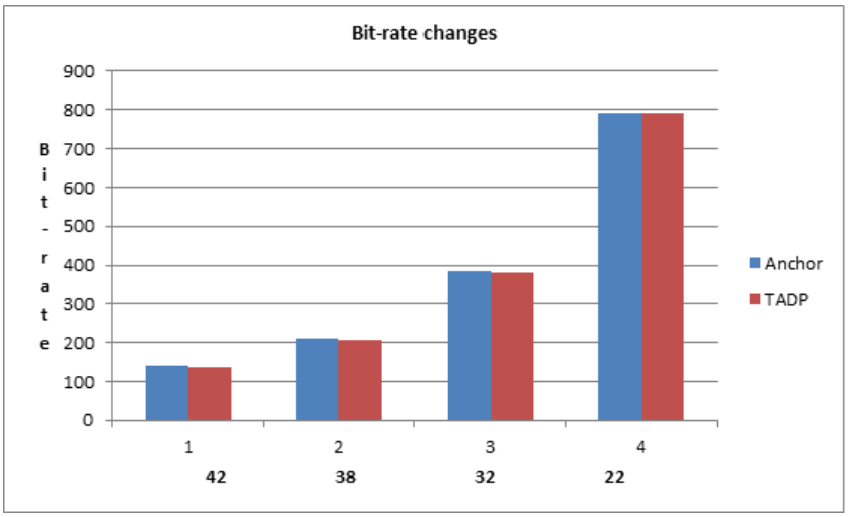

Fig. 4.25: Comparison of the Bit-rate of Akiyo_cif for different QP values

These graphs show bit-rate of the sequences and the difference in the bit-rate can be noticed. In the experiment high contrast, low resolution and sequences with many details are used. Results show the reduction of bit-rate achieved by the proposed algorithm over the original algorithm.

4.3.7 Total running time for the test sequences

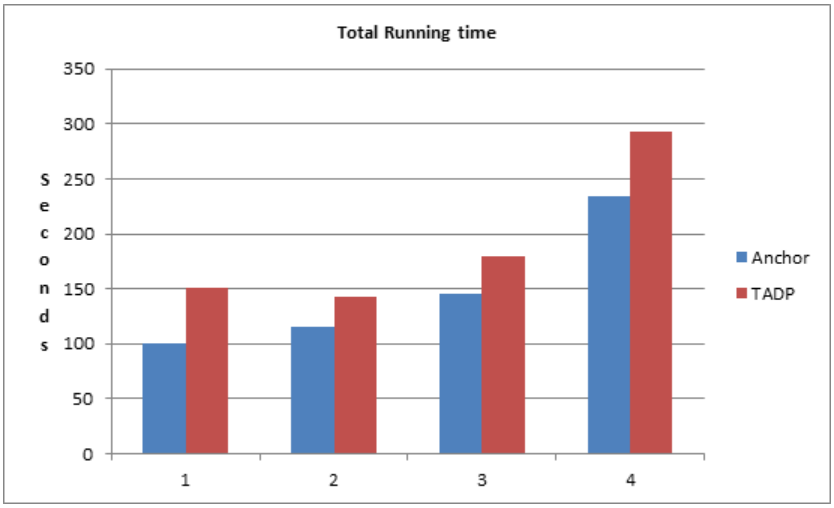

Fig. 4.26: Comparison of the total running time of Bus_cif for different QP values

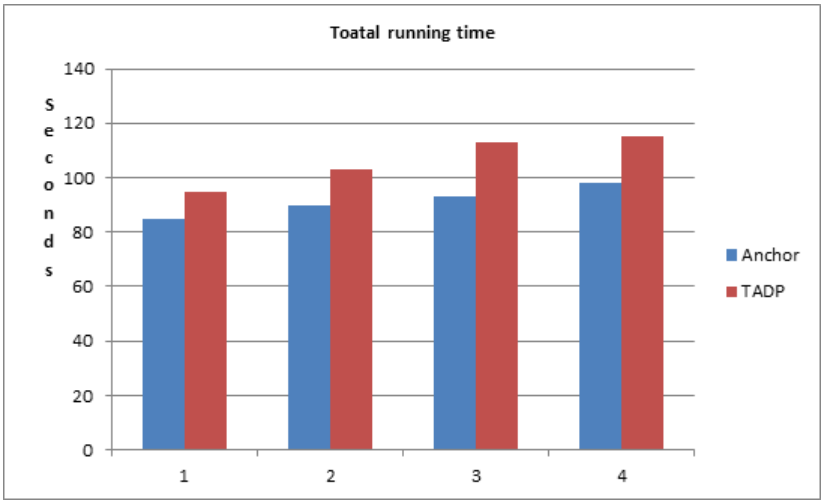

Fig. 4.27: Comparison of the total running time of Akiyo_cif for different QP values

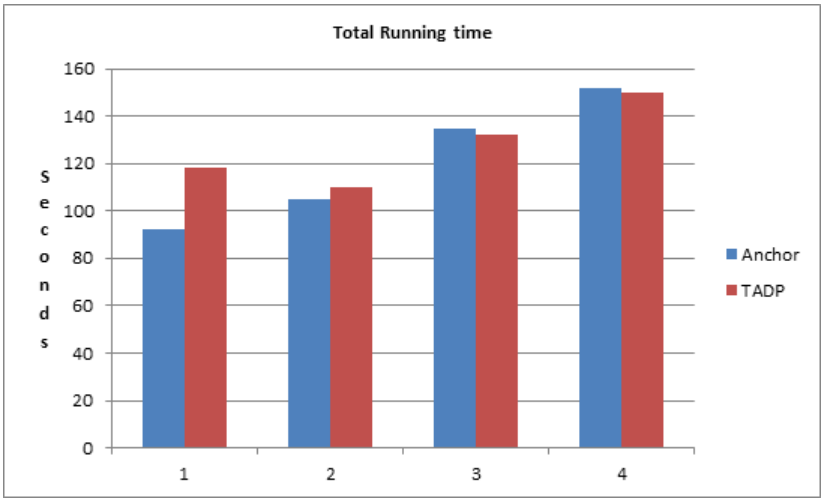

Fig. 4.28: Comparison of the total running time of Mobile_cif for different QP values

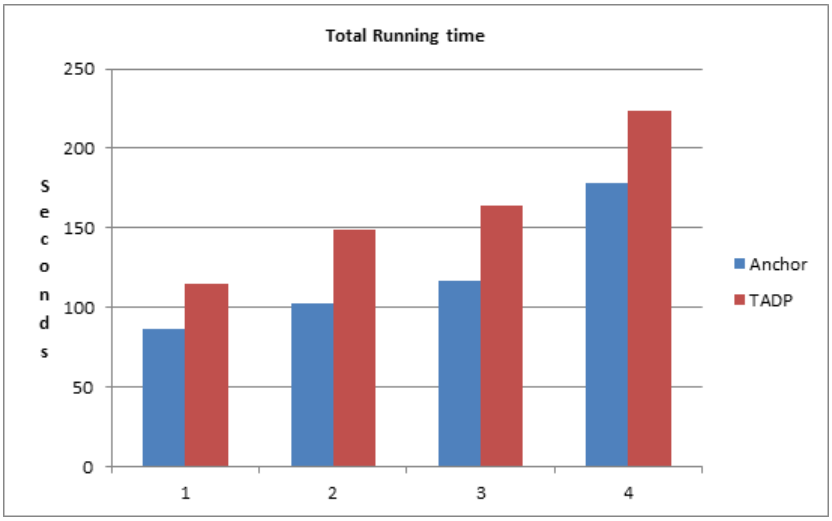

Fig. 4.29: Comparison of the total running time of Foreman_cif for different QP values

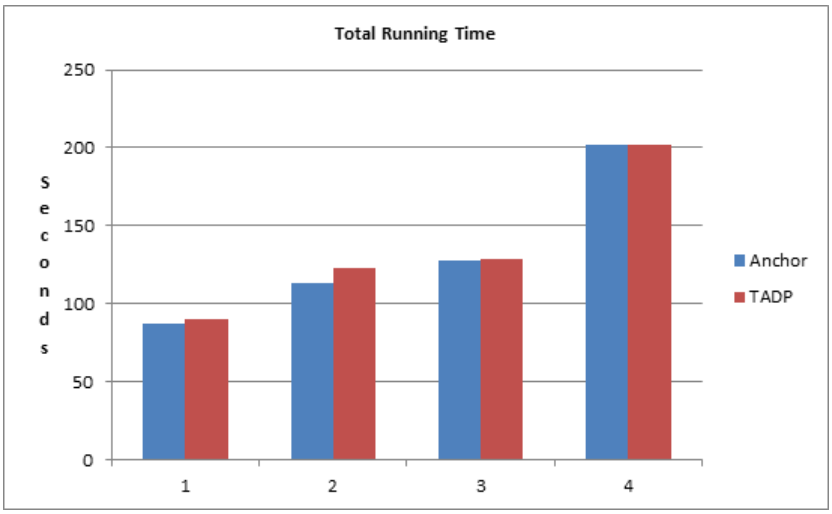

Fig. 4.30: Comparison of the total running time of Flower_cif for different QP values

From the experimental results, we observe that the running time has increased compared to the original algorithm. The increase in total time has varied from sequence to sequence, even though the sequence has more details increase in the total running time is less compared to the sequence with high contrast and static camera. Increase in total time is a bit higher than the original algorithm but the reduction in the bit-rate and increase in the PSNR values reduce the total cost.

4.3.8 Comparison of subjective video quality

Fig. 4.31: (a) Input picture of Flower_cif

Fig. 4.32: (b) Reconstructed picture using original software

Fig. 4.33: (c) Reconstructed picture using proposed software

Fig. 4.34: (a) Input picture of Bus_cif

Fig. 4.35: (b) Reconstructed picture using original software

Fig. 4.36: (c) Reconstructed picture using proposed software

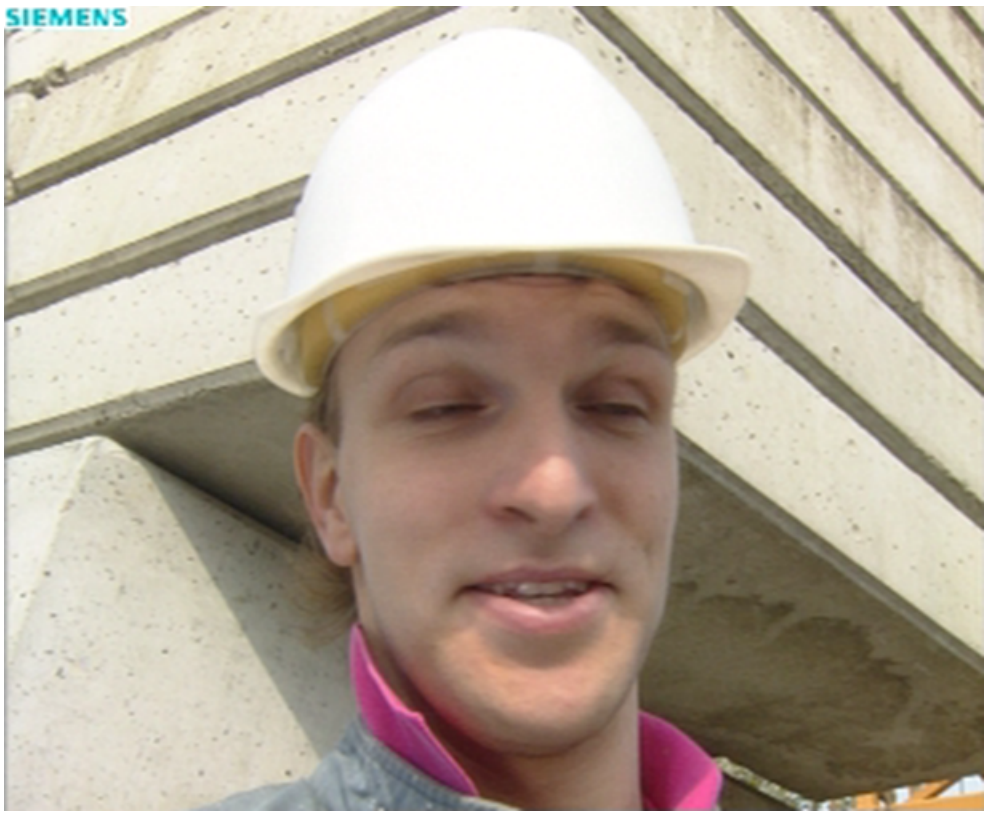

Fig. 4.37: (a) Input picture of Foreman_cif

Fig. 4.38: (b) Reconstructed picture using original software

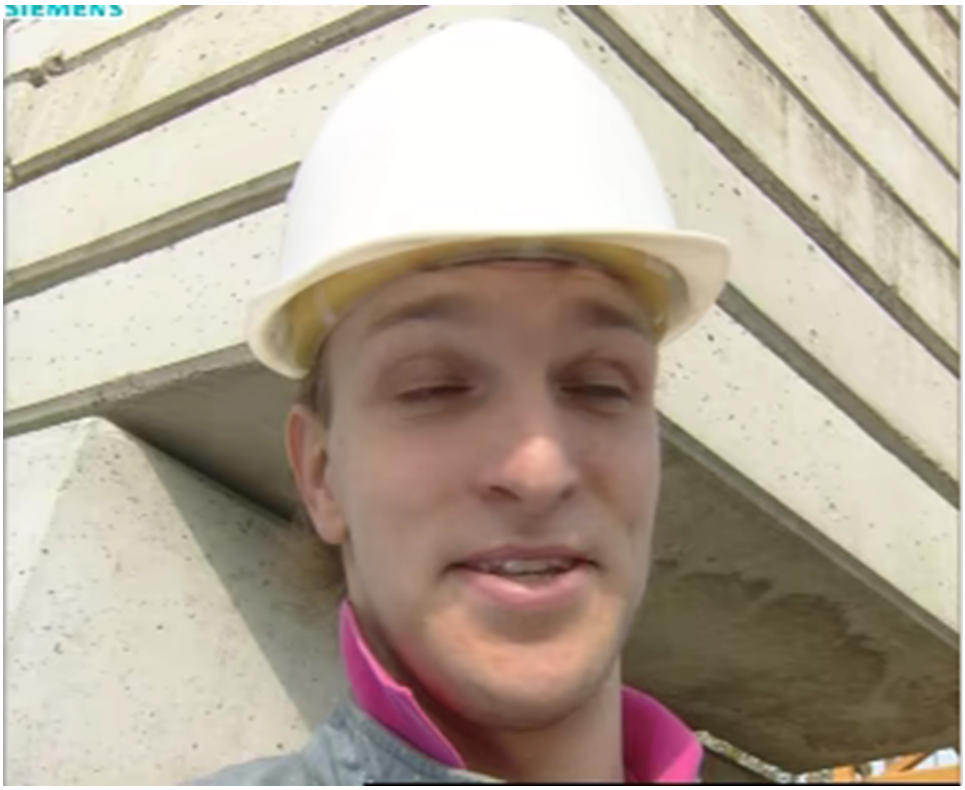

Fig. 4.39: (c) Reconstructed picture using proposed software

Fig. 4.40: (a) Input picture of Mobile_cif

Fig. 4.41: (b) Reconstructed picture using original software

Fig. 4.42: (c) Reconstructed picture using proposed software

Fig. 4.43: (a) Input picture of Akiyo_cif

Fig. 4.44: (b) Reconstructed picture using original software

Fig. 4.45: (c) Reconstructed picture using proposed software

4.4 Summary of test results

The figures in section 4.3 show all the test results, the improved PSNR values and the reduced bit-rate compared to the original software. In this section we give the summarized results of the proposed algorithm and all the details of the original algorithm, these details includes bit-rate, Y-PSNR ,U-PSNR ,V-PSNR and average PSNR and the total time. They are shown in the following tables:

Table 4.2: Changes compared to original software for Mobile_cif

QP values	Bitrate reduced %	YPSNR Increased %	UPSNR Increased %	VPSNR Increased %	Total Time Increased %
22	0.59	0.02	0.09	0.19	25.2
32	0.58	0.03	0.10	0.18	25.3
38	0.56	0.05	0.11	0.17	28.1
42	0.55	0.07	0.13	0.15	28.2

Table 4.3: Changes compared to original software for Akiyo_cif

QP values	Bitrate reduced %	YPSNR Increased %	UPSNR Increased %	VPSNR Increased %	Total Time Increased %
22	0.375	0.055	0.149	0.142	29.931
32	0.37	0.05	0.145	0.141	29.821
38	0.35	0.039	0.138	0.143	29.121
42	0.33	0.035	0.057	0.056	33.704

Table 4.4: Changes compared to original software for Flower_cif

QP values	Bitrate reduced %	YPSNR Increased %	UPSNR Increased %	VPSNR Increased %	Total Time Increased %
22	0.132	0.045	0.097	0.033	20.01
32	0.295	0.088	0.071	0.039	21.09
38	0.261	0.110	0.068	0.023	27.3
42	0.202	0.120	0.056	0.022	28.091

Table 4.5: Changes compared to original software for Foreman_cif

QP values	Bitrate reduced %	YPSNR Increased %	UPSNR Increased %	VPSNR Increased %	Total Time Increased %
22	0.49	0.066	0.133	0.2610	25.84
32	0.35	0.064	0.305	0.3055	27.35
38	0.35	0.054	0.126	0.087	30.03
42	0.32	0.053	0.118	0.086	30.15

Table 4.6: Changes compared to original software for Bus_cif

QP values	Bitrate reduced %	YPSNR Increased %	UPSNR Increased %	VPSNR Increased %	Total Time Increased %
22	0.54	0.024	0.094	0.097	21.08
32	0.38	0.130	0.356	0.108	22.60
38	0.20	0.093	0.019	0.305	23.27
42	0.18	0.142	0.046	0.166	25.05

From the experimental results in table 4.2,4.3,4.4,4.5,4.5 and 4.6 we can see that the proposed algorithm reduce the bit rate by 0.7% and improves the PSNR values by

0.05%, 0.30% and 0.35% in luminance and chroma. As the low resolution sequences usually coded into a relative lower bit rate than the high resolution sequences. So, the improvement is more efficient on low resolution than the high resolution video sequences.

5. SUMMARY

HEVC is designed to achieve multiple goals including coding efficiency, ease of transport system integration and data loss resilience, as well as implementability using parallel processing architecture. In HEVC, there are two processing steps, namely Deblocking Filter (DBF) and Sample Adaptive Offset (SAO). These are applied to the reconstructed samples before writing them to the decoded picture buffer in the decoded loop.

During the development of HEVC, it had also been considered to operate with a third processing step called Adaptive Loop Filter (ALF) after the SAO Filter, however the ALF feature was not included in the final design. In this work we redesigned the in-loop filter by including the ALF. ALF is applied to the reconstructed signal after the deblocking filter and SAO. ALF is used to restore the reconstructed picture such that the mean-squared error between the source picture and the reconstructed picture is minimized.

Additionally, a Trilateral Filter is included to smooth the blocking artifacts of the picture boundary. Trilateral filter and Adaptive loop filters are included in the original in-loop filter. Trilateral filter is applied to block boundaries to improve the perceptual quality and the Adaptive loop filter is used to reduce the bit-rate. Making use of these filters we proposed a new algorithm called A New Adaptive Trilateral Filter for In-loop filtering.

The main aim of proposing A New Adaptive Trilateral Filter for In-loop Filtering is to reduce the bit-rate and improve the PSNR values. We implemented the proposed method on HM-11 and conducted the same experiments using both the modified HM-11 and the original one. For each video sequence varying the Quantization parameters say 22, 32, 38 and 42. Results are noted in both cases, here the input video sequence is in YUV format.

Objective quality and subjective quality are the two types of quality assessment tests performed to evaluate the performance of the proposed algorithm and the original software. The peak signals to noise ratio show the quality of decoded images. The test results show that the proposed algorithm had improved the quality and reduced the bit-rate.

The simulation results show that the proposed algorithm improves rate distortion performance and reduces the ringing artifacts introduced by the use of large transform block sizes and therefore, it also improves the perceived video quality. Moreover, the proposed algorithm allows an average bit-rate reduction of about 0.7% and improves the PSNR of the decoded frame by 0.05%, 0.30% and 0.35% in luminance and chroma.

LIST OF REFERENCES

[1] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The New Gold Standard for Video Compression," *IEEE Consumer Electronics Magazine*, July 2012.

[2] R. Palaparthi, V. K. Srivastava, "A Simple Deblocking Method for Reduction of Blocking Artifacts," *IEEE Students' Conference on Electrical, Electronics and Computer Science*, 2012.

[3] X. Zhang, C. R. Xiong and S. Ma, "Adaptive Loop Filter with Temporal Prediction," *2012 Picture Coding Symposium*, May 7 - 9, 2012, Krakw, Poland.

[4] K. Q. Dinh and h. shim, "Deblocking Filter for Artifact Reduction in Distributed Compressive Video Sensing," VCIP, page 1-5, IEEE 2012.

[5] M. Naccari and F. Pereira, Instituto de Telecomunicaes, "Adaptive Bilateral Filter for Improved In-Loop Filtering in the Emerging High Efficiency Video Coding Standard," *2012 Picture Coding Symposium*, May 7-9, 2012, Krakw, Poland.

[6] C.Y. Tsai, Member, IEEE, C.Y. Chen, T. Yamakage, "Adaptive loop filter for video coding," IEEE Journal of selected topics in signal processing Vol.7 N0. 6, December 2013.

[7] S. Esenlik, M. Narroschke, and T.Wedi, "Syntax refinements for SAO and ALF," *Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G566*, November 2011.

[8] http://www.researchgate.net/publication/37988010_Fast_Trilateral_Filtering/file/d912f50d172f2a9ecf.pdf, Last Date Accessed: March 2014.

[9] http://www.apsipa.org/proceedings_2012/papers/102.pdf, Last Date Accessed: March 2014.

[10] G. J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand, "Overview of the High Efciency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, VOL. 22, NO. 12, December 2012.

[11] C.M. Fu, C.Y. Chen, Y.W. Huang, and S. Lei, "Sample adaptive offset for HEVC," *in Proc. Int. Workshop on Multimedia Signal Processing*, (MMSP), October 2011.

[12] http://codesequoia.wordpress.com/2012/11/04/make-your-first-hevc-stream, Last Date Accessed: March 2014.

[13] P. Lai, F. C. A. Fernandes, H. Guermazi, F. Kossentini, and M.Horowitz, "CE8 Subtest 4: ALF using vertical-size 5 filters with up to 9 coefficients," JCTVC-F303, July 2011.

[14] C.Y. Tsai, C.Y. Chen, C.M. Fu, Y.W. Huang, and S. Lei, "One-pass encoding algorithm for adaptive loop filter in high-efficiency video coding," *Proc. Visual Communications and Image Processing (VCIP)*, November 2011.

[15] http://iphome.hhi.de/suehring/tml/download/KTA/jm11.0kta2.7.zip, Last Date Accessed: March 2014.

[16] T. Yamakage et al., "Description of core experiment 8 (CE8): Nondeblocking loop filtering," Joint Collaborative Team on Video Coding(JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-D608, January 2011.

[17] T. Yamakage, I. S. Chong, and M. Narroschke, "Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-E028," CE8: Non-deblocking loop filteringSummary Report, March 2011.

[18] T. Yamakage, I. S. Chong, and M. Narroschke, "Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-F028," CE8: Non-deblocking Loop FilteringSummary Report, July 2011.

[19] T. Yamakage, I. S. Chong, and M. Narroschke, "Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G038," CE8: Non-Deblocking Loop FilteringSummary Report, November 2011.

[20] C.Y. Chen, C.Y. Tsai, C.M. Fu, Y.W. Huang, and S. Lei, "Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG,JCTVC-G214," Non-CE8: Constrained ALF Coefficients, November 2011.

[21] C.Y. Chen et al., "AHG6: Further Cleanups and Simplifications of the ALF in JCTVC-J0048," Joint Collaborative Team on Video Coding(JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-J0390, July 2012.

[22] C.Y. Chen, C.M. Fu, C.Y. Tsai, Y.W. Huang, and S. Lei, "CE8 Subtest 2: Adaptation between pixel-based and region-based filter selection," Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-E046, March 2011.

[23] C.Y. Chen et al., "Non-CE8.c.7: single-source SAO and ALF virtual boundary processing with cross9 9," Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG,JCTVC-G212, November 2011.

[24] C.Y. Tsai et al., "CE8.d.1: Snowflake5 5 and Cross9 9 for Luma and Chroma ALF Shapes ," Joint Collaborative Team on Video Coding(JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G208, November 2011.

[25] C.Y. Tsai, C.Y. Chen, Y.W. Huang, and S. Lei, "AHG6: ALF with modified padding process," Joint Collaborative Team on Video Coding(JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-J0050, July 2012.

[26] C.Y. Tsai, C.W. Hsu, C.Y. Chen, C.M. Fu, Y.W. Huang, S. Lei,and A. Fuldseth, "AHG4: Non-cross-tiles loop filtering for independent tiles," Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G194, November 2011.